

INTRODUCTION AUX RESEAUX

MODULE NET

3 TC

Créée en 2005
Mise à jour en 2011

Cours
Exercices
Travaux Pratiques

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

FORMATION

Auteur de la Ressource Pédagogique
FRABOULET Antoine

membre de UNIVERSITÉ DE LYON

INSA LYON





Département Télécommunications, services & usages

3TC Module NET :
notes de cours
sujets de TD
sujets de TP

Antoine Fraboulet v.13 2010-2011

Table des matières

Table des figures

Chapitre 1

Présentation

Le module NET se déroule au premier semestre de l'année de 3TC. Ce module permet de découvrir les fondamentaux des réseaux ainsi que leurs illustrations sur la pile TCP/IP utilisée dans l'Internet.

Dans ce module nous présentons en détail les communications au sein des réseaux locaux jusqu'à la première passerelle permettant de sortir du réseau. Le routage des paquets et leur acheminement dans l'Internet, ainsi que les applications, y sont également abordés afin d'avoir une première vue globale de la mise en place des communications grande distance.

Les présentes notes sont un support de base pour le cours, elles contiennent également les énoncés de TD et de TP. Elles ne sauraient être considérées comme complètes sans les notes et compléments que vous pourrez y ajouter lors des cours et des séances de TD/TP. Le sujet d'examen est plus large que les notions abordées dans le document.

Remerciements

- Eric Fleury, INSA de Lyon
- Stéphane Frénot, INSA de Lyon
- Fabrice Valois, INSA de Lyon

Chapitre 2

Introduction

Les présentations des 2 premiers cours sont disponibles sur Moodle à l'adresse http ://moodle.insa-lyon.fr/course/view.php ?id=467

Chapitre 3

Communication dans les réseaux

3.1 Communication dans les réseaux locaux

Un *réseaux local* est un réseau dans lequel toutes les machines sont connectées et peuvent échanger des trames directement (elles sont sur le même support physique).

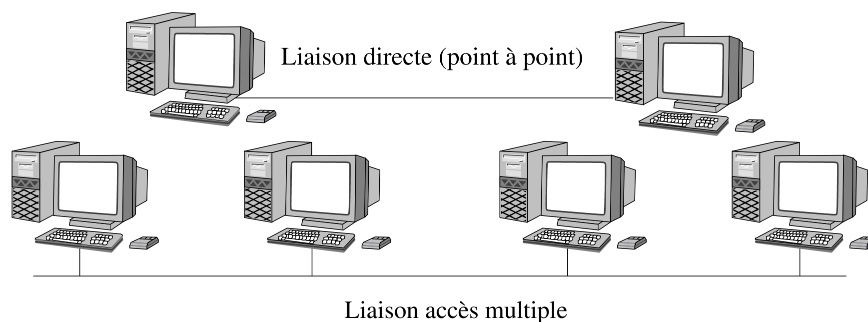


FIGURE 3.1 – Types de communications locales : point à point ou à accès multiples

Les problèmes qui doivent être réglés à ce niveau de communication sont :

- Partage du medium : pour que toutes la machines puissent émettre (partage de canal)
- Gestion des accès concurrents : éviter que toutes les machines émettent en même temps (collisions)
- Equité des accès : toutes les machines doivent pouvoir émettre sans trop attendre (temps de latence)

Ces problèmes sont durs à traiter et souvent ne peuvent être traités tous en même temps. Chaque technologie de réseau local essaye

de résoudre ces problème le mieux possible. Le choix de mise en place d'un réseau local dépend donc de l'importance qu'on accorde aux différents critères. A chaque type de réseau physique on associe un protocole **MAC** (Medium Access Control) qui sera donc en charge de régler les problèmes de partage.

3.1.1 Partage de canal

Il existe plusieurs techniques de partage de canal (que vous pourrez rencontrer plus en détail dans les cours de codage et de modulation par exemple).

FDMA : Multiplexage en fréquence (Frequency Division Multiple Access) : le canal est partagé en bandes de fréquences de façon à ce que plusieurs émetteurs puissent partager le support physique. Ce type est très utilisé en optique par exemple (division selon la longueur d'émission.

TDMA Multiplexage temporel (Time Division Multiple Access) : le canal est unique mais son utilisation est découpée dans le temps. Une tranche de temps (time slot) est alloué pour chaque machine (une machine peut émettre pendant son "time slot" et toutes les autres écoutent). Si une machine ne veut pas émettre le slot est perdu. Ce mode oblige les machines a être synchronisées (à avoir un temps global). Ce point est une importante contrainte technique. Les points forts sont (en autres) : la garantie de débit (en fonction du nombre et de la taille des slots) et l'absence de collision.

CDMA Multiplexage en utilisant des codes orthogonaux (Code Division Multiple Access). Utilisé dans les réseaux sans fils et satellites. Un code unique est attribué à chaque utilisateur. Ces codes sont construits de façon à ce que la réception simultanée de plusieurs messages (provenant de plusieurs utilisateurs) puissent tout de même être décodés sans erreur.

TDMA et FDMA : multiplexage en temps et en fréquence. Ce type de multiplexage est utilisé dans le GSM, par exemple.

3.1.2 Gestion des accès simultanés

La gestion des accès simultanés doit résoudre le problème des machines envoyant des messages en même temps et sur le même canal (donc dans les cas où le partage de canal n'est pas total).

Une gestion assez répandue est la gestion *aléatoire*. Chaque machine peut émettre quand elle veut, au risque de faire une collision. Il

n'y a pas de coordination à priori des machines. Tout est donc une question de probabilité (on retrouve très souvent des probabilités dans les réseaux, ce n'est qu'un début).

En cas de collision, il faut 2 choses : (i) s'en rendre compte et (ii) mettre en place un mécanisme permettant de réémettre les informations en évitant de faire une nouvelle collision – donc arriver à faire communiquer les machines en les décalant dans le temps. Un protocole MAC aléatoire a donc deux choses à spécifier :

- Comment détecter les collisions
- Comment réparer les collisions

Quelques exemples de protocoles aléatoires :

- Aloha
- Slotted Aloha
- CSMA, CSMA/CD, CSMA/CA

3.1.3 Exemple Ethernet

Construit initialement en 1970. Toujours utilisé depuis !

Concepts :

- écouter avant de transmettre (Carrier Sense)
- éviter les collisions avec les transmissions actives (Collision Detection)

En cas de collision, stopper la transmission et attendre une période aléatoire et recommencer l'émission.

CSMA/CD : Carrier Sense Multiple Acces / Collision Detection

- Jam Signal : s'assurer que tous les autres émetteurs ont détecté la collision
- Délais variables : un délai fixe entraîne des collisions
- Si un délai aléatoire avec une espérance fixe est utilisé on peut avoir deux cas :
  - Peu d'émetteur ? attente inutile
  - Trop d'émetteur ? trop de collisions
- But : adapter les tentatives de retransmission pour estimer la charge courante. En cas de charge, l'attente aléatoire sera plus longue.

Calcul du délais / Backoff

- Accroître exponentiellement le délai aléatoire
  - Essayer d'abord avec un délais court.
  - Si on reste en collision on augmente le temps d'attente.
- 1ere collision :

  - Choisir K dans [0 :2^i -1] = {0,1} ;
  - le délai est : K x 512 bit au débit du réseau
- 2nd collision :
  - Choisir K dans {0,1,2,3}
- Pour $i \geq 10$
  - Choisir K dans {0,1,2,3,4,...,1023}

Restriction de la taille du réseau Pour assurer qu'un paquet est transmis sans collision, un hôte doit être en mesure de détecter une collision avant la fin de la transmission. Cela limite la taille du réseau : le temps de propagation sur le réseau doit permettre de faire un **aller/retour pendant le temps d'écoute**. Le temps d'écoute est fixé, pour Ethernet, au temps équivalent à l'envoi de 512 bits à la vitesse nominale. Pour un réseau à 10Mbits/s le temps d'écoute est de **51,2μs**.

3.2 Communication dans les réseaux Ethernet

Les réseaux locaux de type Ethernet permettent d'avoir une communication directe entre les machines qui sont physiquement reliées au réseau. Les trames Ethernet utilisent des adresses physiques inscrites dans les cartes par les constructeurs pour s'identifier. Si on connaît l'adresse (aussi appelée adresse MAC) d'une carte il est possible de lui envoyer des données directement.

Les cartes réseau écoutent les trames qui passent sur le réseau et remontent au système d'exploitation les trames dont l'adresse de destination est celle de la carte ou bien l'adresse de diffusion (**broadcast**). Ce deuxième type d'adresse est utilisé pour envoyer un message à tout le monde (à toutes les machines branchées sur le réseau).

3.2.1 Trame Ethernet

Préambule	Destination	Source	Type	Données	CRC
8	6	6	2	46–1500	4

FIGURE 3.2 – Trame Ethernet

Champs utilisés dans la trame Ethernet :
- Préambule : utilisé pour la synchronisation de la réception de trame.
- Destination : adresse Ethernet de destination
- Source : adresse Ethernet source de la trame

- Type : type de données transportées
- Données : taille maximum 1500 octets. Les données sont complétées par des octets de bourrage pour avoir une taille minimum de 46 octets
- CRC : somme de contrôle sur la trame

Les adresses Ethernet sont composées de 6 octets et sont habituellement notées en hexadécimal sous la forme `12:34:56:78:9a:bc`. Les 3 premiers octets de l'adresse sont fixes pour un constructeur et les 3 derniers servent à assurer l'unicité des adresses physiquement inscrites dans les cartes Ethernet produites en série. L'adresse de broadcast sur Ethernet est `ff:ff:ff:ff:ff:ff`.

Les types rencontrés habituellement pour les trames Ethernet dans un réseau TCP/IP sont les suivantes

- 0x0800 : IPv4
- 0x86DD : IPv6
- 0x0806 : ARP
- liste complète :
  http ://www.iana.org/assignments/ethernet-numbers

Chapitre 4

Les protocoles de l'Internet

4.1 Le protocole IP

IP est un protocole permettant d'avoir une **adresse logique** pour les machines (contrairement aux adresses physiques d'Ethernet et des autres protocoles de couches MAC). Ces adresses sont fixées par l'administrateur du réseau, ce sont donc des adresses qui permettent d'organiser les adresses sur le réseau indépendamment des machines. Une autre particularité des adresses IP est qu'elles sont **globales** : une adresse IP **publique** est censée être unique dans le monde entier.

4.1.1 Trame ARP

Les réseaux locaux sont des réseaux utilisant les adresses physiques des cartes. Par contre, si on ne connaît que l'adresse IP d'une machine il faut pouvoir retrouver son adresse MAC. Le mécanisme de **résolution d'adresse** proposé ici est le protocole ARP (Address Resolution Protocol) qui permet en deux échanges de trames de retrouver l'adresse MAC d'une machine à partir de son adresse IP.

Le protocole ARP est bien un protocole de niveau 3 car il permet au niveau réseau de savoir avec qui il veut communiquer sur le lien Ethernet. ARP est en fait indépendant d'Ethernet et fonctionne avec **n'importe quel type de réseau de niveau 2**. Après la résolution ARP, la couche de niveau 3 ne connaît que la taille des identifiants de niveau 2 et des valeurs numériques pour joindre les machines distantes. De cette façon IP peut fonctionner sur plusieurs types de protocoles de niveau 2 sans changer le principe de base de la résolution d'adresse ARP. Les requêtes ARP sont véhiculées dans des trames de niveau 2.

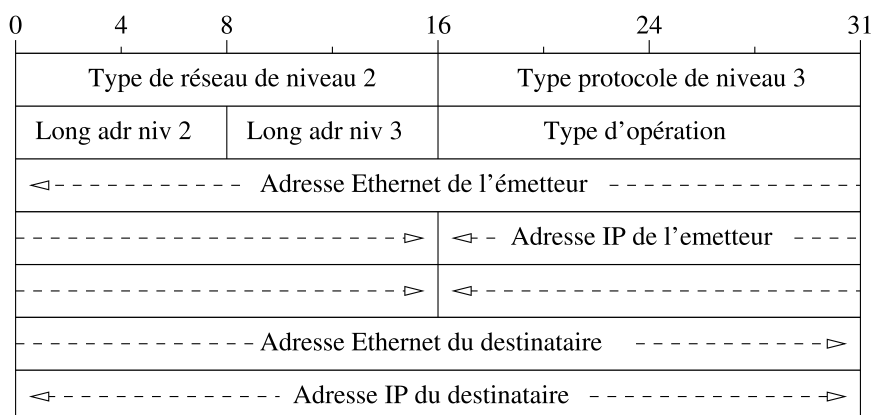


FIGURE 4.1 – Trame ARP

Une requête/réponse ARP est véhiculée par une trame (donc sur Ethernet dans notre configuration) de type 0x0806. La machine émettrice envoie une requête en broadcast (`FF:FF:FF:FF:FF:FF`) afin de contacter toutes les machines du réseau local au niveau Ethernet. La machine qui reconnaît son adresse IP peut renvoyer une réponse ARP en utilisant l'adresse Ethernet inscrite dans la requête comme adresse de destination.

Les traductions IP/ARP sont conservées dans une machine de façon temporaire dans une table ARP. Vous pouvez voir le contenu de la table de la machine sur laquelle vous êtes en tapant la commande (sous Unix) `arp -a`.

ARP ne fonctionne que sur un réseau local. Il n'est pas possible d'utiliser ARP pour commencer une communication avec une machine dans un autre réseau local, même s'il existe un chemin entre les deux. Il n'existe pas d'ARP en dehors d'un réseau local, cela n'a pas de sens de faire cette opération.

4.1.2 En-tête du datagramme IP

Détail des champs de l'en-tête IP :
- Version (4 bits) : indique le numéro de version pour le datagramme IP (4 ou 6)
- HL (4bits) : *Header Length*, taille de l'entête avec les options
- TOS : Type of Service : champs type de service pour IP
- Taille : taille des données du datagramme (16 bits)
- ID : identifiant de datagramme, utilisé pour la fragmentation
- Flags (3 bits) : drapeaux binaires pour la fragmentation (dont bits DF et MF)

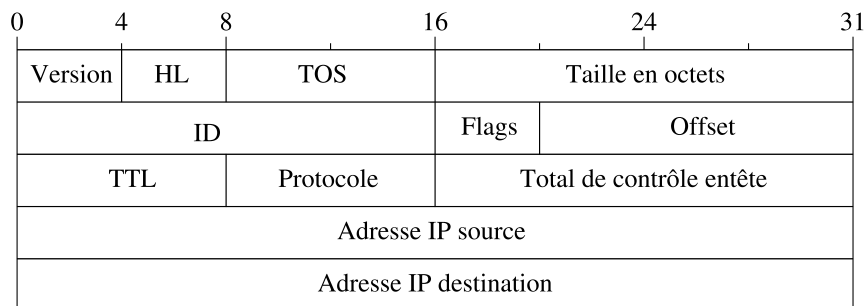


FIGURE 4.2 – En-tête du datagramme IP

- Offset : offset du datagramme courant dans le datagramme fragmenté
- TTL : *time to live* = nombre de sauts maximum sur la route
- Protocole : protocole contenu dans les données IP
- Total de contrôle : somme de contrôle effectuée sur l'entête
- Adresse IP source
- Adresse IP destination

Une liste des protocoles supportés par IP est disponible sur les machines Unix dans le fichier `/etc/protocols`. Dans la suite du cours nous ne nous intéresserons qu'aux protocole ICMP (numéro 1), UDP (numéro 17) et TCP (numéro 6)

4.1.3 Adressage IP

Les adresses IP sont des nombres sur 32 bits dont la représentation usuelle est un groupe de 4 entiers sur 8 bits séparés par des points (*dotted notation*). Ainsi l'adresse 2262191317 (en décimal) correspond à l'adresse 134.214.76.213 dans la notation usuelle. Les adresses IP sont des adresses logiques qui sont fixées par configuration logicielle de la machine. Une machine a au moins autant d'adresses IP que d'interfaces (cartes) réseaux. Elle peut éventuellement en avoir plus.

Les adresses sur Internet sont gérées par l'organisme IANA. C'est l'organisme chargé de répartir les adresses IP dans le monde. Une première liste pour voir à qui appartiennent les tranches d'adresses est disponibles ici

http ://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml.

La gestion est ensuite faite par des organismes régionaux (voir Ripe NCC). La configuration IP d'une machine se résume donc, pour l'instant, aux paramètres suivants :

- adresse IP

– masque de sous réseau
– adresse IP de la passerelle de sortie
– adresse IP d'au moins un serveur DNS (voir partie 6.2)

4.1.4 Décision de routage dans IP

Le réseau local auquel est relié la machine est calculé en effectuant un et logique entre l'adresse IP de la machine et son masque de réseau. Pour savoir si une machine destination est dans le même réseau il faut faire la même opération avec l'adresse que l'on souhaite joindre et son propre masque de réseau. Si le résultat est le même alors les deux machines sont dans le même réseau local et la communication peut s'effectuer via le réseau local. Dans le cas contraire la machine source doit envoyer le datagramme au routeur de sortie (dont elle connaît l'adresse IP par configuration et qu'elle peut joindre en utilisant le LAN). Le routeur se chargera de trouver le prochain saut et ainsi d'acheminer le datagramme un peu plus loin dans le réseau jusqu'à destination. Le nombre maximum de sauts autorisés (passage de routeurs) est fixé au départ par la valeur du champs TTL. Si le TTL arrive à 0 pendant le transport, alors le paquet est détruit et on reçoit en retour un message ICMP de type 11 et de code 0.

ARP ne fonctionne que sur un réseau local. Il n'est pas possible d'utiliser ARP pour commencer une communication avec une machine dans un autre réseau local, même s'il existe un chemin entre les deux. Il n'existe pas d'ARP en dehors d'un réseau local.
Si la machine destination n'est pas dans le **même réseau local** alors la communication passe par la passerelle. Si l'emetteur a besoin de faire un ARP, il ne sera fait que vers l'adresse MAC du routeur de sortie et pas vers la machine destination.

Je sais, c'est la deuxième fois qu'il est mentionné qu'ARP ne fonctionne que dans un réseau local. **C'est parce que c'est important.**

4.1.5 Fragmentation

La fragmentation fait intervenir les champs ID, Flags et Offset de l'entête pour remettre dans l'ordre les morceaux d'un paquet IP fragmenté au cours du routage dans le réseau.

Les routeurs intermédiaires peuvent fragmenter un paquet mais ne font pas la défragmentation. C'est à la machine destinataire d'attendre et de remettre les fragments dans l'ordre. La machine utilise le champs TTL pour mesurer le temps maximum d'attente une fois qu'elle commence à rassembler un datagramme. Si ce TTL passe à 0 alors le datagramme est jeté car les morceaux n'ont pas été reçus à temps.

Le bit DF (*Don't Fragment*) indique que l'on ne souhaite pas qu'un paquet puisse être fragmenté. Si le cas se présente le datagramme est jeté et un message ICMP est renvoyé à la source (type 3, code 4). Ce bit DF permet de faire de la découverte de MTU sur le chemin. Cette "MTU de chemin" est donc le plus gros datagramme que l'on puisse envoyer de bout en bout sans qu'il soit fragmenté (on a donc la MTU la plus petite de tous les réseaux traversés).

4.1.6 Diagnostic sur IP : ICMP

ICMP (Internet Control Message Protocol) est utilisé pour avoir des retours d'information sur l'état du réseau et des machines.

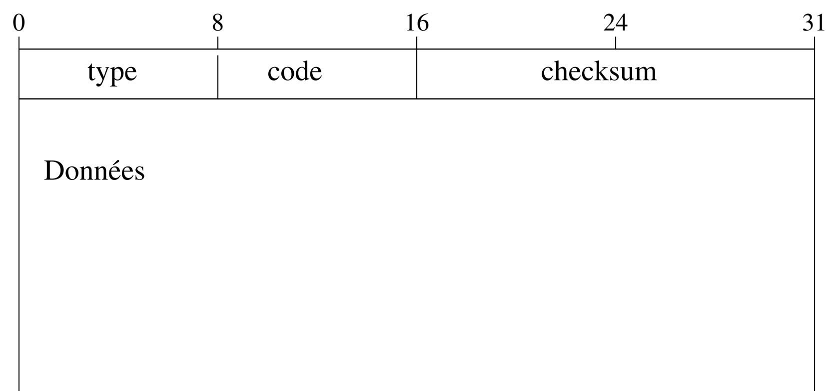


Figure 4.3 – En-tête du datagramme ICMP

Les données, quand il y en a, dépendent du type de paquet ICMP. Si le paquet ICMP est un signalement d'erreurs alors le contenu des données est pris dans les 8 premiers octets des données du datagramme ayant généré l'erreur.

4.1.7 ToS : Type of Service

Le champs ToS permet de donner une information sur la nature des données contenues dans le datagramme IP. Cette information peut être utilisée pour améliorer et diriger le routage. Il faut que

type 0	code=0	echo reply (requête)
type 3		destination unreachable (messages d'erreurs)
	code 0	network unreachable
	code 1	host unreachable
	code 2	protocol unreachable
	code 3	port unreachable
	code 4	fragmentation needed but don't-fragment bit set
	code 5	source route failed
	code 6	destination network unknown
	code 7	destination host unknown
	code 8	source host isolated (obsolete)
	code 9	destination network administratively prohibited
	code 10	destination host administratively prohibited
	code 11	network unreachable for TOS
	code 12	host unreachable for TOS
	code 13	communication administratively prohibited by filtering
	code 14	host precedence violation
	code 15	precedence cutoff in effect
type 4	code=0	source quench (contrôle de flux) (messages d'erreurs)
type 5		redirect (messages d'erreurs)
	code 0	redirect for network
	code 1	redirect for host
	code 2	redirect for type-of-service and network
	code 3	redirect for type-of-service and host
type 8	code=0	echo request (ping) (requête)
type 9	code=0	router advertisement (requête)
type 10	code=0	router solicitation (requête)
type 11		TTL (messages d'erreurs)
	code 0	TTL = 0 during transit
	code 1	TTL = 0 during reassembly
type 12		parameter problem (messages d'erreurs)
	code 0	IP header bad
	code 1	required option missing
type 13	code=0	timestamp request
type 14	code=0	timestamp reply
type 17	code=0	address mask request
type 18	code=0	address mask reply

Table 4.1 – Table des types et codes ICMP

les routeurs sur le chemin soient configurés pour en tenir compte. En pratique très peu de routeurs acceptent de prendre en compte le champs ce qui le rend inutilisable.

4.1.8 Protocoles de transports

IP permet de communiquer avec une *machine*. Pour atteindre un service (application) il faut donc plus de précision dans l'adresse logique du destinataire du datagramme.

Les protocoles de transport UDP et TCP font l'interface entre le réseau (comment atteindre une machine) et les applications (comment atteindre un programme précis qui tourne dans une machine).

De la même manière que pour les adresses IP (il faut connaître une adresse IP à l'avance pour communiquer avec une machine), il faut connaître le port (numéro d'accès au service ou programme) pour accéder à un service particulier sur un serveur. Il existe pour cela une liste de référence connue sous le nom de "well known ports" qui est la même pour toutes les machines. Il s'agit ici d'établir une communication de **bout en bout**. Les deux protocoles utilisés sur l'Internet sont UDP et TCP.

4.2 UDP : User Datagram Protocol

UDP est un protocole en mode datagramme et propose seulement le service de multiplexage applicatif par rapport à IP. UDP ne propose pas de service supplémentaire et on reste donc dans un mode non connecté avec des problèmes potentiels de perte de paquets dans le réseau (ce ne sera pas le cas de TCP).

UDP reste néanmoins un protocole rapide qui est très efficace et est utilisé par de nombreuses applications dont le service d'annuaire DNS (port 53)

- port source : port de communication (socket) ouvert par l'émetteur.
- port destination : port de communication côté serveur (destinataire)
- taille : taille en octets
- somme de contrôle : somme de contrôle de l'entête UDP

4.2.1 Propriétés des datagramme UDP

UDP utilise IP comme support, il est donc soumis aux mêmes contraintes de fiabilité :

- support de transmission non fiable

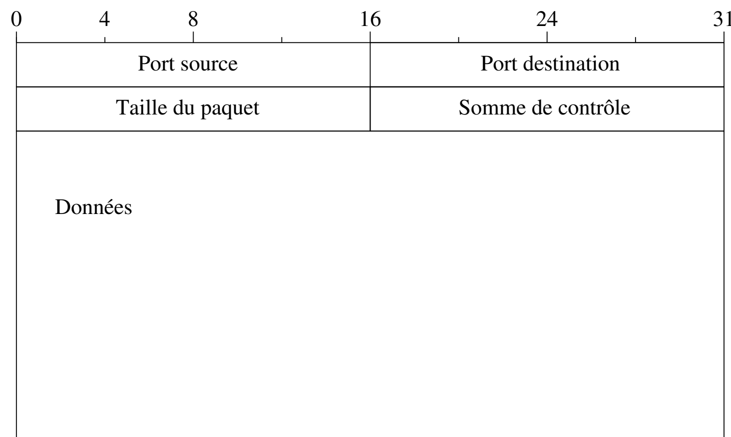


FIGURE 4.4 – Structure du datagramme UDP

– perte de datagrammes possibles
– l'ordre des datagrammes peut être inversé pendant le trajet sur le réseau (s'ils ne prennent pas la même route)

La taille maximale d'un datagramme UDP est limité par la taille des paquets IP. Les paquets UDP sont indépendants. Pour acheminer une information dont la taille dépasserait celle

Remarque importante : Les données sont envoyées dès que l'application effectue une écriture. Chaque écriture de l'application génère un datagramme UDP. Les lectures des datagrammes depuis l'application se font sur des datagrammes complets.

4.3 TCP : Transmission Control Protocol

TCP permet d'avoir une connexion entre programmes ayant des propriétés bien plus complexes que les datagrammes UDP. TCP propose, par l'intermédiaire d'acquittements, d'avoir un contrôle de perte de paquet avec livraison des données à l'application dans l'ordre d'envoi ainsi qu'un contrôle de congestion du réseau.

– TCP temporise les données envoyées sous forme de segments dont la taille est adaptée aux conditions présentes sur le réseau
– Chaque segment est acquitté par le destinataire pour avoir un transport fiable
– La perte de paquets est contrôlée à l'aide de temporisations
– Les données sont transmises "en ordre" à l'application
– TCP propose un mécanisme de fenêtrage pour ne pas saturer le mémoire de l'application

– Les connexions TCP sont bidirectionnelles.

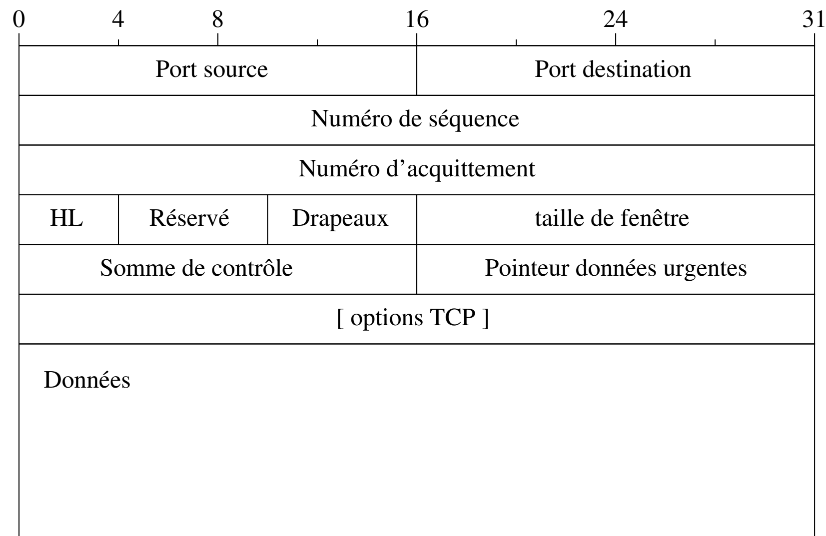


FIGURE 4.5 – Structure du segment TCP

- port source : port TCP sur la machine source
- port destination : port TCP sur la machine destination
- numéro de séquence : numéro du premier octet du segment
- numéro d'acquittement : numéro du premier octet attendu
- HL : Header Length, permet de savoir s'il y a des options dans l'entête, la taille de l'entête est indiquée en nombre de mots de 32 bits (comme pour le champs HL du datagramme IP)
- Réservé : bits réservés pour un usage futur
- Drapeaux :
  - URG : le pointeur de données urgente est valide
  - ACK : le numéro d'acquittement est valide
  - PSH : push, le receveur doit passer les données le plus rapidement
  - RST : reset, réinitialisation de la connexion
  - SYN : synchronisation de numéro de séquence, début de connexion
  - FIN : signale que l'émetteur a fini d'émettre
- taille de fenêtre : place disponible dans la mémoire de réception de l'annonceur
- somme de contrôle : somme de contrôle appliquée sur l'entête et les données du segment
- pointeur données urgentes :

Les numéros de séquence et d'acquittement identifient le nombre d'octets transmis sur la connexion. Le compteur ne commence pas de

zéro mais du nombre annoncé lors de la synchronisation d'ouverture de connexion (ISN : Initial Sequence Number).

Une communication TCP est identifiée par le quadruplet :

<adresse IP source, port source, adresse IP destination, port destination>

L'option la plus courante dans TCP est l'annonce de la taille maximum de segment (MSS : Maximum Segment Size). Cette option est présente dans les segments ayant le bit SYN positionné pour indiqué quelle est la quantité maximale de données que veut recevoir l'émetteur. Lorsque la connexion est locale le MSS est fixé à la MTU du protocole de niveau 2 moins la taille des entêtes IP et TCP : pour un réseau Ethernet le MSS sera fixé à 1460 dans la plupart des cas (cela peut dépendre des systèmes d'exploitation) sinon, pour une connexion distante, le MSS est fixé à 536 (la plus petite MTU sur Internet est fixée à 576 octets).

Les acquittements ne sont pas envoyés dès que l'on reçoit des données. Une temporisation de 500ms (au maximum) est mise en place pour augmenter la probabilité d'avoir des données à transmettre en même temps que l'acquittement et ainsi augmenter le débit utile du réseau.

Le mécanisme d'ouverture de connexion TCP est prévu pour fonctionner dans le cas où les deux machines font une ouverture simultanée (cas très rare). Dans ce cas, une seule connexion TCP bidirectionnelle résulte de cette double ouverture.

4.3.1 États d'une connexion TCP

- rouge : côté serveur
- bleu : connexion active (côté client). Une socket dans l'état LISTEN peut émettre des données et initier une connexion mais le cas reste très rare.
- magenta : ouverture simultanée, les deux côté font une ouverture active (cas extrêmement rare)
- vert : fermeture active et fermeture passive
- rose : fermeture simultanée (la fin de connexion se fait en 3 étapes au lieu de 4)
- noir : fermeture de la socket par l'application alors que la connexion n'est pas encore établie

L'état `TIME_WAIT` est appelé état d'attente 2`MSL`. Un `MSL` est la durée de vie maximum d'un segment, cette valeur est comprise entre 30s et 2 min. C'est la durée maximale pendant laquelle un segment peut exister sur le réseau avant d'être rejeté.

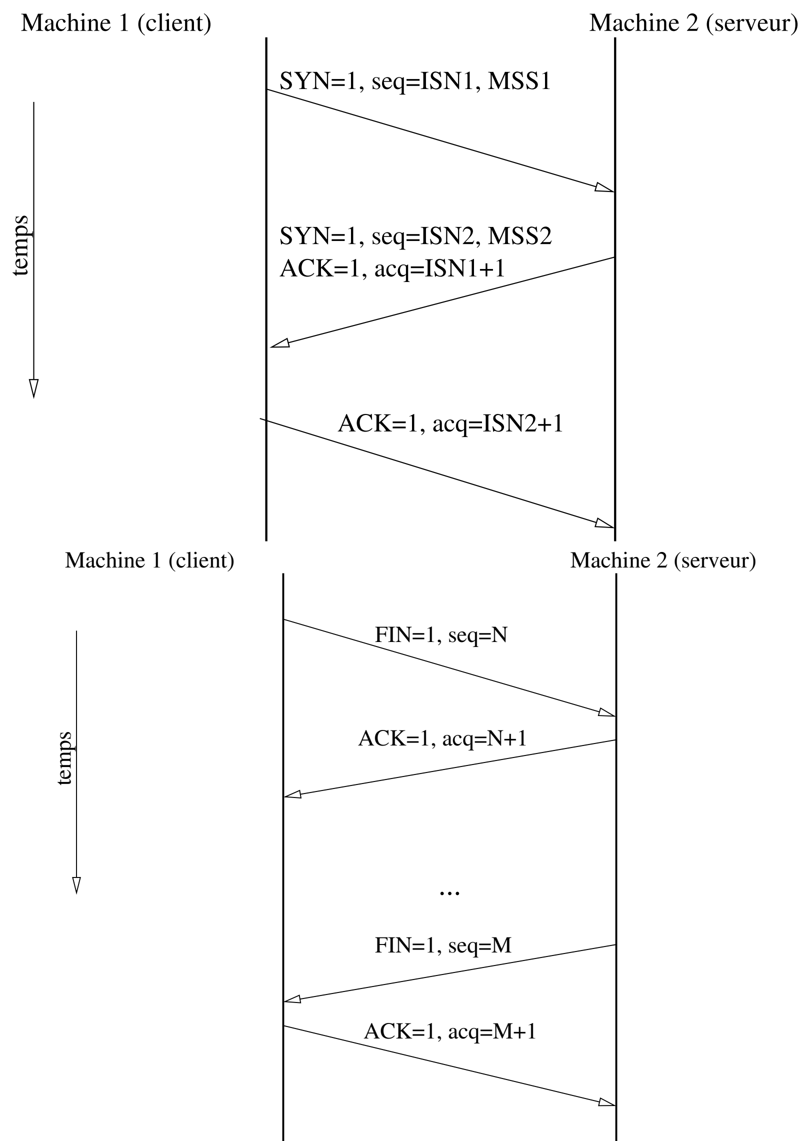


Figure 4.6 – Établissement et fermeture de connexion TCP

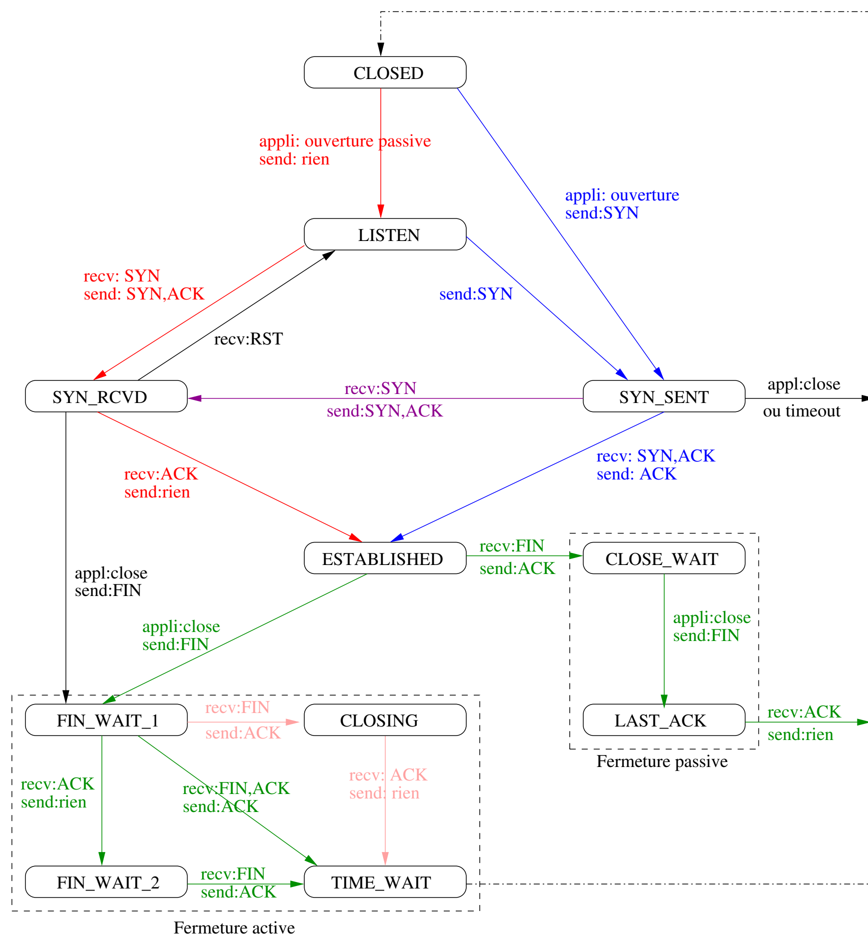


FIGURE 4.7 – États d'une connexion TCP

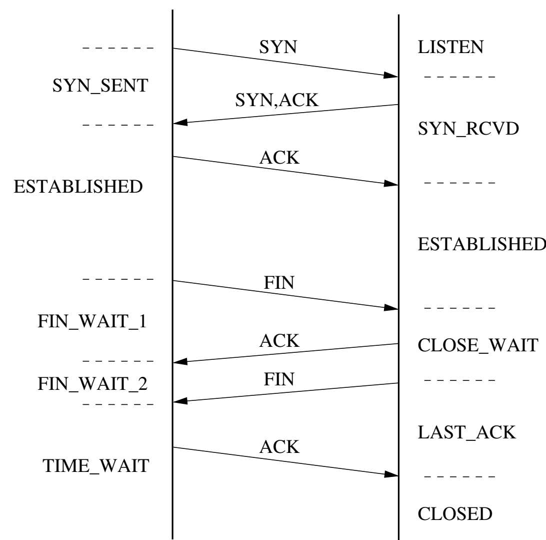


Figure 4.8 – États d'une connexion TCP dans le temps

Dans le cas d'une fermeture simultanée, la partie gauche et la partie droite passent par les états `FIN_WAIT_1`, `CLOSING` et `TIME_WAIT` au lieu de passer par les états mentionnés sur la figure précédente.

TCP propose des mécanismes pour désynchroniser les envois de segment des acquittements. Chaque machine annonce la quantité de mémoire disponible dans son buffer de réception (champs window dans l'entête TCP). L'émetteur sait alors quelle quantité de données il peut espérer envoyer sans saturer le destinataire. La taille de cette fenêtre peut varier de 8Ko à 16Ko (ou plus) selon les systèmes.

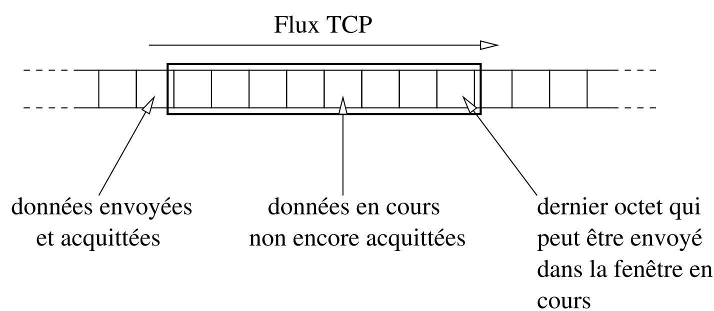


Figure 4.9 – Fenêtre glissante de TCP

Une application qui aurait indiqué une fenêtre vide dans son dernier acquittement doit réémettre un segment avec le même numéro d'acquittement que le précédent pour réanoncer sa nouvelle taille de fenêtre.

L'envoi des segments ainsi que l'envoi des acquittements peut être différé par rapport aux données envoyées et lues de l'application (algorithmes de Clark et Nagle vus en TD)

4.3.2 Contrôle de congestion

Le contrôle de congestion sert à adapter le débit de TCP par rapport aux pertes de paquets pouvant arriver dans le réseau. L'adaptation du débit dans TCP est appelé slow start.

La fenêtre de réception permet à l'émetteur d'envoyer rapidement plusieurs segments TCP sans attendre d'acquittement entre l'envoi des segments. Ce mécanisme fonctionne très bien sur un même LAN mais peut poser des problème en cas de congestion du réseau.

TCP possède une fenêtre de gestion de la congestion appelée congestion window (cwnd). Cette fenêtre est locale à l'émetteur et sa valeur n'est pas inscrite dans l'entête de segment. Alors que la fenêtre annoncée dans les entêtes TCP conditionne la quantité de données traitées côté récepteur, la fenêtre de congestion conditionne la quantité de données qui peut être envoyée côté émetteur.

L'émetteur peut émettre à hauteur du minimum entre la taille de la fenêtre annoncée et la taille de la fenêtre de congestion.

L'émetteur commence à émettre un segment et attend l'acquittement. Une fois l'acquittement reçu, l'émetteur peut alors émettre deux segments et attendre les acquittements. La taille de la fenêtre de congestion est agrandie de façon exponentielle jusqu'à trouver la limite de congestion du réseau. Une fois cette limite atteinte, des segments commencent à se perdre et l'émetteur retombe alors à une fenêtre de congestion à 1. Il recommence le "slow start" exponentiel jusqu'au précédent point de rupture. Une fois le point de rupture atteint l'augmentation devient linéaire jusqu'au prochain seuil de rupture (et ainsi de suite).

Le diagramme standard pour représenter l'avancée d'un flux TCP est un diagramme Temps/numéro de séquence. Une émission sans perte se traduit par un numéro de séquence croissant (sauf passage au delà de la limite de taille du compteur tous les 2^{32} octets (4Go). Des pertes de segments TCP se traduise par des reprises à des numéros de segment déjà envoyés quelques instant auparavant.

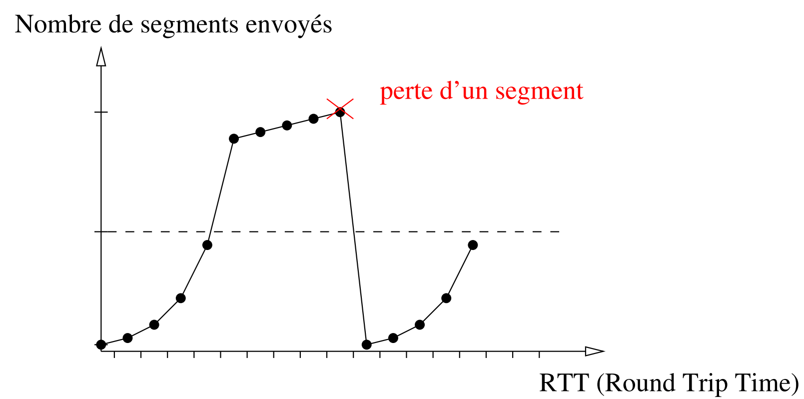


FIGURE 4.10 – Démarrage lent de TCP : slow start

Chapitre 5

Programmation, l'API Socket

5.1 Programmation de l'API socket en C

5.1.1 Introduction

Une socket est une interface de communication introduite par les systèmes Unix pour la communication réseau. Il s'agit d'un point d'accès aux services de la couche transport, c'est-à-dire TCP ou UDP. La communication par sockets sur un réseau adopte généralement un modèle client-serveur ; en d'autres termes pour communiquer il faut créer un serveur prêt à recevoir les requêtes d'un client.

Dans tous les cas, avant d'utiliser une socket il faut la **créer**, c'est-à-dire créer un descripteur associé à l'ensemble des informations constituant la socket (buffers, adresse, port, état, etc.). Ensuite il est éventuellement possible de l'**attacher à une adresse locale** représentant la provenance des messages envoyés.

Côté serveur : la création de la socket est suivie d'une mise en **attente** de message dans le cas d'une communication UDP, ou de mise en attente de connexion dans le cas d'une communication TCP. Dans le cas d'une communication TCP, il est généralement profitable de permettre au serveur de gérer plusieurs connexions simultanées ; dans ce cas un nouveau processus sera créé pour chaque connexion ou bien les sockets seront enregistrées pour être utilisées dans un appel système `select(2)`.

Côté client : la communication se fait tout d'abord en renseignant l'adresse du serveur à contacter. Ensuite peut avoir lieu l'envoi proprement dit de données ou la demande de connexion (selon le cas).

Ce document constituant seulement une introduction à la programmation par sockets, il est volontairement simplificateur sur la

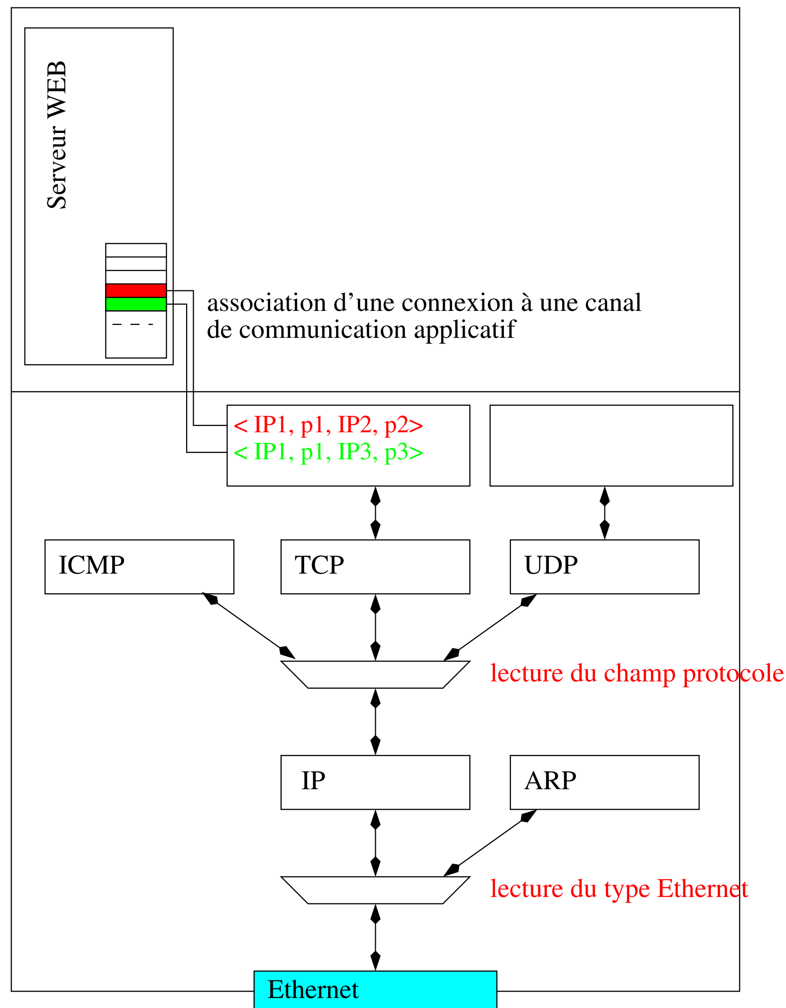


FIGURE 5.1 – Pile TCP/IP : l'API socket

plupart des concepts et ne présente seulement les fonctions les plus importantes.

Client UDP

1 Création de socket (socket())
2 Attribution de l'adresse locale (bind())
3 Envoie d'un paquet (write() ou sendto())
4 Attente, eventuelle, de la réponse (read() ou recvfrom())

Serveur UDP

1 Création de socket (socket())
2 Attribution de l'adresse locale (bind())
3 Boucle (souvent infinie)
  3.1 Lecture d'un datagramme (read() ou recvfrom())
    [*Traitement*]
  3.2 Envoie de la réponse (write() ou sendto())

5.1.2 Création d'une socket

La création d'une socket se fait par l'appel système `socket(2)` dont la déclaration se trouve dans `<sys/socket.h>`. Cet appel permet de créer une structure en mémoire contenant tous les renseignements associés à la socket (buffers, adresse, etc.) ; il renvoie un descripteur de fichier permettant d'identifier la socket créée (-1 en cas d'erreur).

```
int socket (
  int domain,   /* AF_INET pour l'internet                   */
  int type,     /* SOCK_DGRAM pour une communication UDP,    */
                /*   SOCK_STREAM pour une communication TCP */
  int protocole /* 0 pour le protocole par defaut du type    */
);
```

Une fois la socket créée, il est possible de lui attacher une adresse qui sera généralement l'adresse locale ; sans adresse une socket ne pourra pas être contactée (il s'agit simplement d'une structure qui ne peut pas être vue de l'extérieur). L'attachement permet de préciser l'adresse ainsi que le port de la socket. On attache une adresse à une socket à l'aide de la fonction `bind(2)` qui renvoie 0 en cas de succès et -1 sinon.

```
int bind (
  int descr,              /* descripteur de la socket */
  struct sockaddr *addr, /* adresse a attacher       */
  int addr_size           /* taille de l'adresse      */
);
```

Exemple : Cet exemple définit une fonction permettant de créer une socket et de l'attacher sur le port spécifié de l'hôte local.

```
/* ****************************************************************
 * type : type de la socket a creer = SOCK_DGRAM ou SOCK_STREAM
 * port : numéro de port désiré pour l'attachement en local
 *****************************************************************/
int creer_socket (int type, int port)
{
  int desc;
  int longueur=sizeof(struct sockaddr_in);
  struct sockaddr_in adresse;

  /* Creation de la socket */
  if ((desc=socket(AF_INET,type,0)) == -1)
  {
    perror("Creation de socket impossible");
    return -1;
  }

  /* Preparation de l'adresse d'attachement = adresse IP Internet */
  adresse.sin_family=AF_INET;

  /* Indication de l'adresse IP locale de la socket       */
  /* Conversion (interne) -> (reseau) avec htonl et htons */
  /* INADDR_ANY: toutes les interfaces présentes          */
  adresse.sin_addr.s_addr=htonl(INADDR_ANY);

  /* Indication du port local de la socket                */
  /* si port = 0, l'adresse sera choisie au hasard par    */
  /* le système au dessus de 1024                         */
  adresse.sin_port=htons(port);

  /* Demande d'attachement de la socket */
  if (bind(desc,(struct sockaddr*)&adresse,longueur) == -1)
  {
    perror("Attachement de la socket impossible");
    close(desc);
    return -1;
  }

  /* Pour récupérer les informations d'attachement local on
   * peut utiliser la fonction getsockname(2)
   *
   *   struct sockaddr_in adresse;
   *   getsockname(desc,(struct sockaddr*)&adresse,&longueur);
   */

  return desc;
}
```

5.1.3 Communication UDP par envoi de message

Afin d'établir une communication UDP entre deux machines, il faut d'une part créer un serveur sur la machine réceptrice et d'autre part créer un client sur la machine émettrice. Ensuite la communication peut se faire à l'aide des fonctions `sendto(2)` et `recvfrom(2)`. Chaque utilisation de `sendto` génère un paquet UDP qui doit être lu en une seule fois par la fonction `recvfrom`.

UDP Côté serveur

Il s'agit ici de créer la socket qui recevra le message. Ensuite on attend le message à l'aide de la fonction `recvfrom`.

```
int recvfrom (
  int desc,                     /* descripteur de la socket   */
  void *message,                /* adresse de reception       */
  int longueur,                 /* taille de la zone reservee */
  int option,                   /* 0 pour une lecture simple  */
  struct sockaddr *ptr_adresse, /* adresse emetteur           */
  int *long_adresse             /* taille de la zone adresse  */
);
```

Exemple : On illustre ici le côté serveur par la création d'un processus permettant la réception d'un unique message sur le port passé en argument.

```
[...]

int main (int argc, char *argv[])
{
  struct sockaddr_in adresse;
  int port,desc_socket,lg=sizeof(adresse);
  char message[4096];

  if (argc < 2)
    {
      fprintf(stderr,"udp_serveur num_socket\n");
      return 1;
    }
  /* creation et attachement de la socket */
  port=atoi(argv[1]);
  if ((desc_socket=creer_socket(SOCK_DGRAM, port)) == -1)
  {
    fprintf(stderr,"Creation de socket impossible\n");
    exit(2);
  }
  /* attente du message */
  recvfrom(desc_socket,message,4096,0,
                    (struct sockaddr*)&adresse,&lg);
  /* &adresse contient les informations sur l'émetteur */
  printf("Message : %s", message);
  close(desc_socket);
  return 0;
}
```

UDP Côté client

Il s'agit ici d'énvoyer un message sur une machine distante. Pour cela on commence par créer la socket émettrice, puis on prépare l'adresse de destination et on envoie le message à l'aide de la fonction `sendto`.

```
int sendto (
  int desc,                         /* descripteur de la socket  */
  void *message,                    /* message a envoyer         */
  int longueur,                     /* longueur du message       */
  int option,                       /* 0 pour un envoi simple    */
  struct sockaddr *ptr_adresse,     /* adresse destinataire      */
  int *long_adresse                 /* taille de la zone adresse */
);
```

Exemple : On illustre ici le côté client par la creation d'un processus permettant l'énvoi du message "Salut" sur la machine et le port specifié en argument.

```
[...]

int main (int argc, char *argv[])
{
  struct sockaddr_in adresse;
  int port,desc_socket,lg=sizeof(adresse);
  struct hostent *hp;
  char message[]="Salut\n";

  if (argc < 3)
  {
    fprintf(stderr,"udp_client machine port_distant\n");
    exit(1);
  }
  /* creation et attachement socket sur un port quelconque */
  port=0;
  if ((desc_socket=creer_socket(SOCK_DGRAM, &port)) == -1)
  {
    fprintf(stderr,"Creation de socket impossible\n");
    exit(2);
  }
  /* recherche de l'adresse internet du serveur */
  if ((hp=gethostbyname(argv[1])) == NULL)
  {
    fprintf(stderr,"Machine %s inconnue\n",argv[1]);
    exit(3);
  }

  /* preparation de l'adresse destinatrice */
  adresse.sin_family=AF_INET;
  adresse.sin_port=htons(atoi(argv[2]));
  memcpy(&adresse.sin_addr.s_addr,hp->h_addr,hp->h_length);

  /* envoi du message */
  sendto(desc_socket,message,strlen(message)+1,0,
              (struct sockaddr*)&adresse,lg);
  close(desc_socket);
  return 0;
}
```

5.1.4 Communication TCP

Afin d'établir une communication TCP entre deux machines, il faut d'une part créer un serveur sur la machine réceptrice, d'autre part créer un client sur la machine émettrice. Il faut ensuite réaliser une connexion entre les deux machines, qui sera gérée côté serveur par les fonctions `listen(2)` et `accept(2)`, et côté client par la fonction `connect(2)`. La communication peut alors se faire à l'aide des fonctions `write(2)` et `read(2)`.

TCP Côté serveur

Il s'agit ici de créer la socket qui acceptera la connexion. On déclare alors la socket comme *acceptant* les connexions à l'aide de la fonction listen (retourne 0 en cas de succès).

```
int listen (
  int desc,        /* descripteur de la socket */
  int nb_pendantes /* nombre maximal de connexions en attente */
);
```

Afin d'attendre une nouvelle demande de connexion, on utilise la fonction accept(). A l'arrivée d'une nouvelle demande de connexion, cette fonction retourne un descripteur correspondant à une nouvelle socket créée pour l'occasion (ou -1 si une erreur s'est produite). La communication est alors possible par l'intermédiaire de cette dernière socket, qui s'utilise comme un fichier de caractères (ou un tube).

```
int accept (
  int desc,                     /* descripteur de la socket */
  struct sockaddr *ptr_adresse, /* adresse de lémeteur */
  int *long_adresse             /* taille de la zone adresse */
);
```

Si l'on souhaite alléger la charge du serveur, il est également possible de créer un nouveau processus gérant cette communication afin de permettre au serveur de retourner immédiatement à un état d'attente de demande de connexion.

Exemple : On illustre ici le côté serveur par la création d'un processus permettant la réception de connexions sur le port passé en argument, et l'écho de tous les caractères envoyés lors d'une connexion.

```
[...]

int main (int argc, char *argv[])
{
  struct sockaddr_in adresse;
  int port,socket_ecoute,socket_service,lg=sizeof(adresse);
  char car;

  if (argc < 2) {
    fprintf(stderr,"udp_serveur num_socket\n");
    return 1;
  }
  /* creation et attachement de la socket */
  port=atoi(argv[1]);
  if ((socket_ecoute=creer_socket(SOCK_STREAM, port)) == -1) {
    fprintf(stderr,"Creation de socket impossible\n");
    exit(2);
  }
  /* declaration de l'ouverture du service */
  if (listen(socket_ecoute, 10) == -1) {
    perror("Listen");
    exit(1);
  }
  /* boucle de prise en charge des connexions        */
  /* on accepte ici qu'une seule connexion à la fois */
  while (1) {
    socket_service=accept(socket_ecoute,(struct sockaddr*)
               &adresse, &lg);
    if (socket_service == -1) {
      perror("Accept");
      exit(3);
    }
    /* la connexion est acceptee, on lit tout */
    while (read(socket_service, &car, sizeof(car))) {
      write(socket_service, &car, sizeof(car));
      if (car == 'x')
        break;
    }
    /* femeture de la socket créée par le accept(), la socket
     * socket_ecoute reste active */
    close(socket_service);
  }
}
```

TCP Coté client

Il s'agit ici d'établir une connexion avec une machine distante afin de pouvoir communiquer par l'envoi de flux de caratères. Pour cela on commence par créer la socket émettrice, puis on prépare l'adresse de destination avant de faire la demande de connexion à l'aide de la fonction `connect(2)` (qui retourne -1 sur erreur).

```
int connect (
  int desc,                     /* descripteur de la socket */
  struct sockaddr *ptr_adresse, /* adresse du destinataire */
  int *long_adresse             /* taille de la zone adresse */
);
```

Exemple : On illustre ici le côté client par la création d'un processus permettant une connexion puis l'envoi du message "salut" sur la machine et le port specifiés en argument. Ce client lit également les messages retournés par le serveur.

```
[...]

int main (int argc, char *argv[])
{
  struct sockaddr_in adresse;
  int port,desc_socket,lg=sizeof(adresse);
  struct hostent *hp;
  char message[]="Salut\n",car;

  /* controle du nombre de parametres */
  if (argc < 3) {
    fprintf(stderr,"tcp_client machine port\n");
    exit(1);
  }

  /* creation et attachement sur un port quelconque */
  port=0;
  if ((desc_socket=creer_socket(SOCK_STREAM, port)) == -1)
  {
    fprintf(stderr,"Creation de socket impossible\n");
    exit(2);
  }

  /* recherche de l'adresse internet du serveur */
  if ((hp = gethostbyname(argv[1])) == NULL)
  {
    fprintf(stderr,"Machine %s inconnue\n",argv[1]);
    exit(3);
  }

  /* preparation de l'adresse destinatrice */
  adresse.sin_family=AF_INET;
  adresse.sin_port=htons(atoi(argv[2]));
  memcpy(&(adresse.sin_addr.s_addr), hp->h_addr, hp->h_length);
  /* demande de connexion au serveur */
  if (connect(desc_socket,(struct sockaddr*) &adresse, lg) == -1)
  {
    perror ("Connect");
    exit(4);
  }
  write(desc_socket, message, strlen(message));
  car='x';
  write(desc_socket, &car, sizeof(char));
  while (read(desc_socket, &car, sizeof(char))) {
    printf("%c",car);
  }
  printf("\n");
  close(desc_socket);
  return 0;
}
```

5.1.5 Pour aller plus loin

Il existe de nombreux autres cas d'utilisation des sockets ainsi qu'une multitude de façons d'utiliser et de combiner les fonctions d'accès au réseau. Les pages du manuel Unix (`man`) vous seront d'une aide précieuse pour aller au delà de ces simples exemples. Un des livres de référence pour la programmation réseau sous Unix est celui de Richard Stevens [4].

Chapitre 6

Exemples d'applications

6.1 DHCP

`DHCP` (Dynamic Host Configuration Protocol) (RFC 2131) est un protocole permettant d'automatiser la configuration des machines sur un réseau IP.

1. Lorsqu'une machine démarre, elle ne connaît pas son adresse IP. La seule chose dont elle dispose est son adresse MAC Ethernet (la plupart du temps). Si un client DHCP est configuré sur cette machine, il va essayer d'envoyer des requêtes dans le réseau local pour savoir si une machine (un serveur DHCP) peut lui envoyer une configuration IP.
2. Le ou les serveurs sur le réseau répond(ent) en envoyant leur adresse IP et une proposition de bail. Le client prend la première réponse qui lui arrive.
3. Le client répond alors au serveur pour valider le bail qui lui a été proposé.
4. Le serveur confirme le bail. L'adresse allouée ne sera plus proposée à un autre client pour toute la durée du bail.

- Les serveurs DHCP renvoient une adresse libre dans une plage d'adresses qui leur a été assignée. L'adresse peut donc changer à chaque connexion.
- Les serveurs DHCP permettent également d'avoir des configurations statiques en renvoyant une adresse fixe en fonction de l'adresse MAC de la machine faisant la demande.
- La configuration peut être couplée avec un DNS dynamique pour avoir un enregistrement dans le DNS pendant la durée du bail.

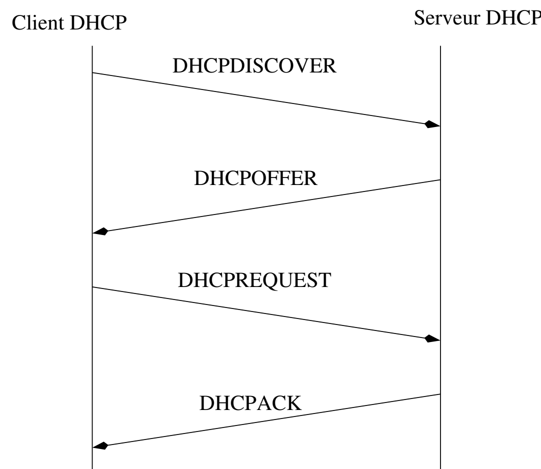


Figure 6.1 – Autoconfiguration par DHCP

– Une autre version de protocole d'auto configuration, BOOTP, permet à une machine de démarrer sans système d'exploitation résidant en local. L'image du système d'exploitation ainsi que tous les fichiers associés sont envoyés par le réseau à la machine pour qu'elle puisse démarrer.

6.2 DNS : Domain Name System

Le service de résolution de nom entre en jeu à chaque fois que l'on veut communiquer avec une machine dont on connaît le nom qualifié mais pas l'adresse IP. Il existe deux types de requêtes DNS : les requêtes *itératives* et les requêtes récursives.

L'organisation des DNS est hiérarchique et les zones (ou domaines) sont gérés en local par les administrateurs. Trouver une adresse à partir d'un nom de machine revient donc à trouver les serveurs successifs à contacter jusqu'au serveur de nom final qui connaîtra la conversion exacte entre nom et adresse IP.

Les résolutions peuvent porter sur le domaine local, auquel cas le serveur de nom répondra directement, ou bien porter sur des noms complets (FQDN : Fully Qualified Domain Name) indiquant le nom de la machine ainsi que le domaine dans lequel elle se trouve.

Les requêtes récursives sont une demande de résolution complète et l'on attend en retour une réponse complète qui peut être le numéro IP de la machine ou l'inexistence de la machine. C'est le DNS à qui l'on fait une demande qui se charge de parcourir l'Internet à la recherche de la machine.

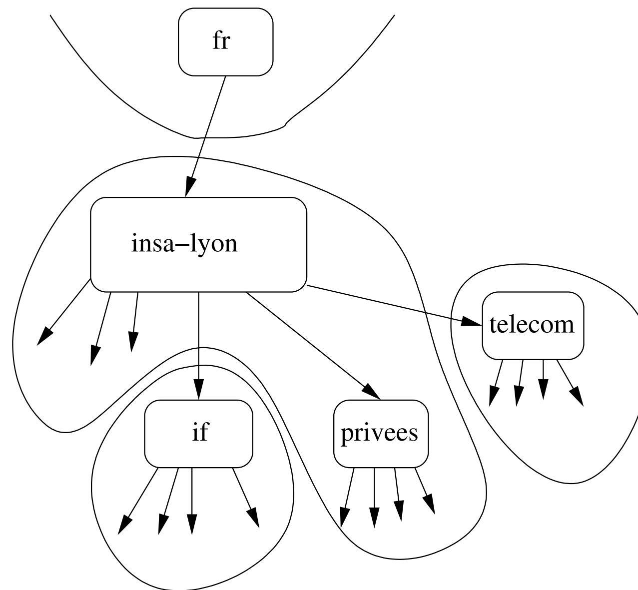


Figure 6.2 – Zones de délégation des noms et des adresses

Il n'existe qu'un type de paquet pour les questions et les réponses DNS. Le message a une entête fixe de 12 octets et 4 champs de taille variable. L'identifiant est positionné par le client et retourné par le serveur. Cela permet au client de reconnaître les réponses aux questions qu'il a envoyé.

0 – 15	16 – 31
Identification	flags
nombre de questions	nombre de réponse RR
nombre de RR avec autorité	nombre de RR additionnels
questions	
réponses	
autorités	
informations additionnelles	

Figure 6.3 – Structure d'une requête DNS

Les drapeaux représentent plusieurs champs :

– QR : 1 bit : 0 si le message est une requête, 1 sinon

– opcode : 4 bits : valeur normale à 0 (question standard). 1 pour

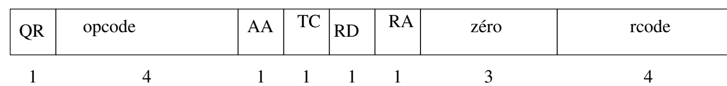


Figure 6.4 – Structure des drapeaux d'une requête DNS

les requêtes inverses. 2 pour les demandes d'état d'un serveur.
- AA : 1 bit : "authoritative answer." Le DNS est autorité pour le domaine en question.
- TC : 1 bit : "truncated." Avec UDP cela indique que la réponse était plus grande que 512 octets et que seuls les 512 premiers octets ont été conservés.
- RD : 1 bit : "recursion desired." Ce bit peut être placé dans les questions et dans les réponses. Il indique au serveur de traiter la requête lui même de façon récursive. Si le bit n'est pas mis et que le serveur n'a pas de réponse avec autorité, la requête est renvoyée au client avec une liste de serveurs à contacter pour aller plus loin.
- RA : 1-bit : "recursion available." Ce bit est mis à 1 si le serveur supporte la récursion. Presque tous les serveurs supportent la récursions, sauf certains serveurs racines.
- zéro : 3 bits : fixés à 0
- rcode : 4-bit : code de retour. Le code 0 indique qu'il n'y a pas eu d'erreurs. Le code 3 est une erreur de nom (name error). Une erreur de nom est renvoyé par un serveur autorité et indique que le domaine n'existe pas.

Requêtes itératives, requêtes récursives

Dans l'exemple de la figure précédente, le serveur de nom X qui reçoit la requête récursive du résolveur ne connaît ni la machine tc-nx.insa-lyon.fr, ni les domaines insa-lyon.fr et fr.

C'est le serveurX qui va se charger de gérer la recherche de la machine avec une succession de requêtes itératives. Toutes les demandes sont faites sur le nom complet. Un serveur de nom renvoit le maximum de l'information dont il dispose. Cela est rendu possible grâce au système de cache. Une première requête est lancée sur un serveur racine pour se renseigner sur l'adresse. Le serveur racine connaît les DNS du domaine fr mais ne possède pas celles de insa-lyon.fr, il renvoie juste l'adresse d'un des DNS de fr.

Un DNS renvoie l'information la plus complète possible dans l'état de ses connaissances (zones dont il est propriétaire ainsi que sa mémoire cache dans laquelle il garde une trace de toutes les adresses qu'il voit passer).

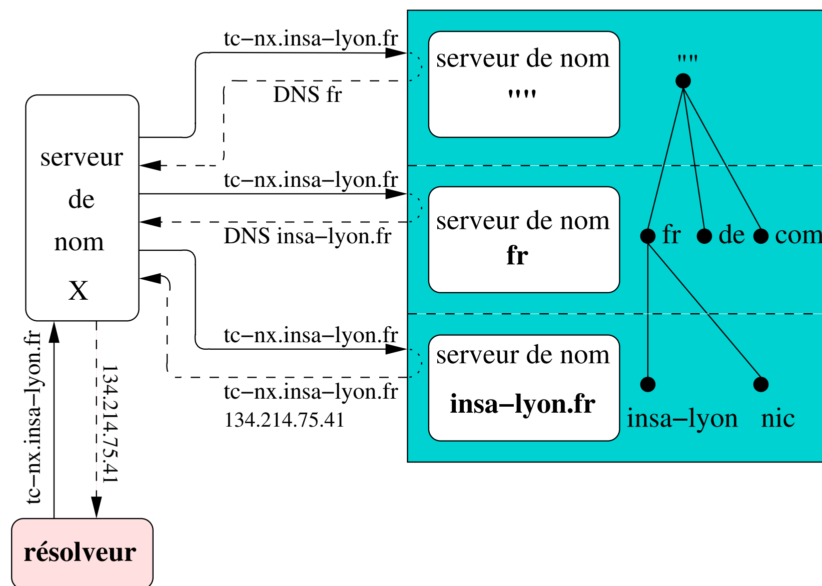


FIGURE 6.5 – Gestion des requêtes DNS

Le serveur de nom X poursuit sa recherche en utilisant le serveur de fr renvoyé par la racine. Celui-ci ne connaît toujours pas l'adresse complète mais peut renvoyer la liste des serveurs de insa-lyon.fr. La dernière requête sur insa-lyon.fr permet de retrouver enfin le numéro IP de la machine (ou de dire qu'elle n'existe pas).

De manière générale, on peut voir le système DNS comme la plus grande base de données répartie au monde. On peut effectuer des requêtes sur des serveurs et c'est au demandeur d'effectuer les traitements adéquats.

Le logiciel BIND (Berkeley Internet Name Domain) propose les outils pour mettre en place sur des machines Unix le système de DNS.

Ce logiciel comprend :

- un serveur DNS : named(8)
- une bibliothèque C pour la résolution DNS : resolver(3)
- des outils pour s'assurer du bon fonctionnement d'un DNS.

La partie résolveur est incluse en standard dans quasiment toutes les bibliothèques C Unix, l'installation d'un service DNS correspond uniquement à l'installation et à la configuration d'un serveur `named`. Dans la programmation sous Unix on peut utiliser le résolveur par les appels systèmes gethostbyname(2) et gethostbyaddr(2). Le premier permet de faire la correspondance nom—adresse IP alors que le second fait la résolution inverse.

Un client DNS est disponible sous Windows nativement. Des ser-

veurs DNS sont aussi disponibles. Nous n'allons aborder dans le TP la partie serveur que sous Unix car c'est celle que l'on rencontre sur la très grande majorité des machines.

Configuration

La partie cliente du système de résolution de nom est configurée sur chaque machine d'un réseau. C'est cette partie qui gère les demandes d'adresses de tous les programmes.

Configuration statique des adresses

Un fichier est utilisé pour faire une configuration statique des noms des machines. Sous Unix ce fichier s'appelle `/etc/hosts`, il est également disponible sous Windows.

```
#adresse_IP nom_qualifié alias1 alias2 ...
127.0.0.1 localhost
192.168.1.1 tc-res-40.insa-lyon.fr tc-res-40
```

Ce système est employé pour les très petits réseaux qui ne sont pas reliés à l'Internet. En effet, le fichier `hosts` doit être mis à jour sur chaque machine à chaque changement et ne permet pas de trouver des machines à partir d'un site distant.

Configuration dynamique

Le passage à un service de nommage dynamique permet de supprimer complètement la méthode utilisant un fichier sur chaque machine.

Sous Unix la configuration d'un client se fait par l'intermédiaire du fichier `/etc/resolv.conf`. La configuration sous Windows est placée dans une interface graphique (dans les propriétés du protocole IP).

Ce fichier est en fait le fichier de configuration de la partie cliente du service de nommage. Il est utilisé à chaque fois que l'on veut résoudre une adresse. On y trouve généralement le nom de domaine de la machine, les noms de domaines par lesquels on peut compléter une adresse ainsi que les adresses IP des DNS qui acceptent des demandes.

```
domain insa-lyon.fr
search univ-lyon1.fr
nameserver 134.214.100.245
nameserver 134.214.100.6
```

Choix de la méthode de recherche

On peut mélanger différents systèmes de résolution de nom (statique, DNS). Le fichier `/etc/host.conf` permet de décider la procédure à suivre pour rechercher une machine. On y trouve principalement la ligne suivante :

```
order hosts,bind
```

Cet ordre indique que lors d'une résolution la recherche commence par la table stockée dans `/etc/hosts` et si l'information n'est pas disponible de faire une requête DNS.

Serveur DNS : configuration

Un serveur DNS peut être configuré sur n'importe quelle machine Unix ou NT. Le démon qui gère le serveur DNS sous Unix s'appelle named et fait partie du paquetage BIND, c'est ce logiciel que nous allons utiliser pour le TP (bind version 8).

`named` utilise les informations de `named.conf` lors de son démarrage pour charger les tables de configurations de ses zones.

```
// This is the primary configuration file for the BIND DNS
// server named.
//
// Please read /usr/share/doc/bind/README.Debian for information
// on the structure of BIND configuration files in Debian for
// BIND versions 8.2.1 and later, *BEFORE* you customize this
// configuration file.

options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers
        // you want to talk to, you might need to uncomment
        // the query-source directive below. Previous versions of
        // BIND always asked questions using port 53, but BIND
        // 8.1 and later use an unprivileged port by default.

        // query-source address * port 53;

        listen-on {
               127.0.0.1;
               192.168/16;
               192.168.2/8;
        };

        // If your ISP provided one or more IP addresses for
        // stable nameservers, you probably want to use them
        // as forwarders. Uncomment the following block, and
        // insert the addresses replacing
        // the all-0's placeholder.

        forwarders {
                212.27.32.176;
                212.27.32.177;
        };
};

// reduce log verbosity on issues outside our control
logging {
        category lame-servers { null; };
        category cname { null; };
};

// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/etc/bind/db.root";
};
```

[... fin du fichier sur la page suivante ...]

```
            [... suite de la page précédente ...]

// be authoritative for the localhost forward and reverse zones,
// and for broadcast zones as per RFC 1912
zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};

zone "insa-lyon.fr" {
        type master;
        file "/etc/bind/db.insalyon";
};
```

Ces fichiers contiennent les informations suivantes : répertoire où sont stockés les fichiers de configuration de chaque zone (un par zone) du DNS ainsi que les zones dont le DNS s'occupe. La zone "." de type *hint* est particulière, elle comprend tous les DNS racine de l'Internet.

Pour chaque zone décrite, on doit avoir un fichier de résolution directe et un fichier de résolution inverse du domaine `in-addr.arpa` grâce auquel on peut retrouver le nom d'une machine par son numéro IP.

Une zone correspond à la liste des machines dont le nom est géré par le DNS. Si une machine est dans le domaine d'un DNS mais pas dans sa zone alors il y a délégation et on passe par un DNS intermédiaire propriétaire de la zone. Exemple de fichier de zone :

```
insa-lyon.fr   IN   SOA   dns.insa-lyon.fr root.insa-lyon.fr. (
                              1        ; Serial
                         604800        ; Refresh
                          86400        ; Retry
                        2419200        ; Expire
                         604800 )      ; Negative Cache TTL
;
@       IN      NS      dns.insa-lyon.fr.
dns     IN      A       134.214.100.6
tc-nx   IN      A       134.214.75.41
[...]
```

Une adresse pleinement qualifiée doit ici être terminée par un point (comme `dns.insa-lyon.fr.`) pour éviter toute confusion et la transformer en dns.insa-lyon.fr.insa-lyon.fr..

Il s'agit, ici, de la zone de résolution directe. Les lignes sont interprétées de gauche à droite, on y trouve le nom de la machine, le protocole de nom associé, le type de renseignement et enfin ces renseignements.

Traduction des champs :

- IN : est le protocole utilisé, ici Internet.
- SOA : (Start Of Authority) indique des renseignements sur la zone : dans l'ordre le nom du DNS sur lequel on est, l'adresse mail de son mainteneur et les temps de vie des informations du DNS
- Serial : est le numéro de version du fichier, il sert aux DNS secondaires pour savoir quand recharger une zone
- Refresh : laps de temps au bout duquel un serveur secondaire doit s'assurer qu'il est à jour (24 heures)
- Retry : temps à laisser passer avant de recommencer si le refresh n'a pas pu être fait (2 heures)
- Expire : temps au bout duquel un secondaire considère les données comme invalide s'il n'arrive plus à contacter son serveur primaire (30 jours)
- Default TTL : temps pendant lequel la réponse de ce serveur doit être considérée comme pertinente (Time To Live)

Type de renseignements que l'on peut trouver :

- NS : Serveur de noms pour le domaine demandé
- A : Adresse IP de la machine demandée
- CNAME : Nom réel de la machine (alias)
- MX : serveur de mail où renvoyer le courrier si la machine est indisponible
- RP : personne responsable de la machine
- TXT : texte de description que l'on peut associer à une machine

DNS racines

Il existe une liste de serveurs de noms connus servant de racine pour les requêtes DNS. Les DNS des TLD (Top Level Domain) doivent s'enregistrer auprès des 13 DNS racines pour pouvoir être trouvés lors des requêtes. Les DNS racines sont répartis à travers le monde et assurent le service de nommage de l'Internet.

```
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
[...]
;
; housed in Japan, operated by WIDE
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

Chapitre 7

Utilisation avancée

7.1 Utilisation et configuration avancée

7.1.1 Architecture des équipements des réseaux

Les routeurs, ou plus généralement les machines ayant un rôle dans un réseau peuvent être vues comme des nœuds d'interconnexion. La représentation usuelle contient toujours des mémoires d'entrées, une interconnexion et des mémoires de sorties.

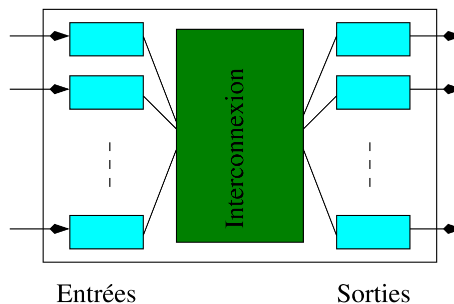


FIGURE 7.1 – Passage des informations dans un routeur

7.1.2 Filtrage

Les *Firewall* sont des équipements de routage qui ont une fonctionnalité supplémentaire pouvant autoriser ou non la retransmission des datagrammes. Le filtrage peut s'effectuer sur les champs des entêtes de paquets (IP, TCP, UDP, ...). Plus le filtrage a lieu dans des couches hautes de protocoles, plus il coûte cher en temps de traitement.

Dans le schéma précédent, on peut décider de faire remonter les trames dans un routeur jusqu'au niveau souhaité pour les examiner

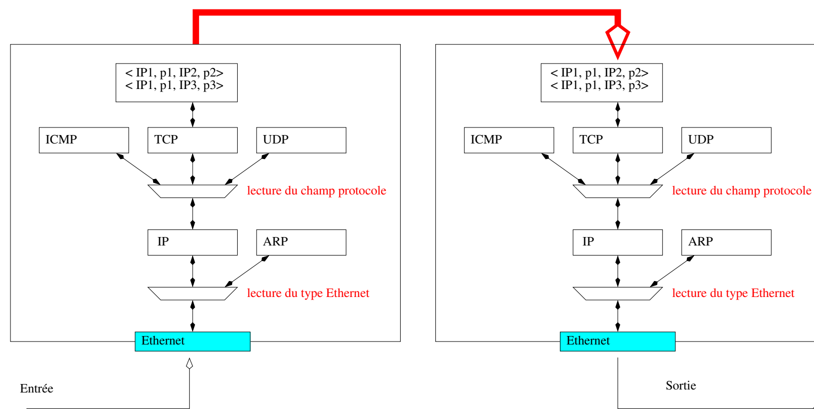


FIGURE 7.2 – Filtrage de paquets IP

avant de les renvoyer sur l'interface de sortie (après avoir regardé la table de routage pour trouver le prochain saut).

En règle générale un filtre est utilisé en entrée d'un réseau pour limiter et contrôler le trafic entrant. Pour mettre en place un filtrage efficace il faut prendre soin de regarder et adapter les règles **dans les deux sens** de communication.

Les règles de filtrage sont en général construites pour ne rien laisser passer sauf les services dont on a besoin (la règle par défaut est donc de tout rejeter). Il faut faire attention à ne pas bloquer une machine complètement sous peine de ne plus pouvoir communiquer. Il faut pouvoir distinguer les réponses aux requêtes émises depuis l'intérieur d'un réseau des demandes de connexions initiées de l'extérieur.

7.1.3 Filtrage des connexions TCP

Le filtrage de connexion TCP est possible si l'on autorise la lecture des bits de signalisation de TCP. Ainsi, on peut filtrer un début de connexion pour l'interdire si la demande provient de l'extérieur d'un réseau et le laisser passer si l'émetteur est à l'intérieur. Cette mise en place nécessite une table dynamique dans la machine faisant office de filtre pour qu'elle puisse se rappeler qui a initié la connexion lors du passage d'un paquet. Pour les connexions TCP cette table recense les quadruplets TCP au fur et à mesure de l'établissement des connexions. Les entrées dans la table sont enlevées lors du passage d'un segment avec le bit FIN à 1.

Ce mécanisme est extensibles pour filtrer de façon sélective les connexions UDP, ICMP, ...

7.1.4 Translation d'adresse

Bien que les machines effectuant de la translation d'adresses puissent être vues comme des routeurs (les datagrammes les traversent) les mécanismes de translation d'adresse doivent être différenciés du routage. En effet, la translation de paquets modifie les adresses sources et/ou destination ainsi que les ports des datagrammes, ce qui est interdit pour les routeurs.

On distingue deux types de translation :
- translation d'adresse source : SNAT
- translation d'adresse destination : DNAT

Le SNAT est utilisé pour permettre à un réseau adressé dans une tranche d'adresse privée à sortir et à avoir des communications IP avec l'Internet. Dans la translation d'adresse source les champs d'adresse IP source et de port source pour les connexions UDP et TCP sont modifiés. De façon analogue au filtrage, la machine faisant le NAT doit conserver une table de correspondance pour faire l'opération inverse (modification de l'adresse et du port destination) lors du passage du paquet de réponse.

Le DNAT est utilisé pour permettre à une machine de l'Internet d'utiliser un service sur une machine étant dans une tranche d'adresse privée. On parle parfois de serveur virtuel dans le sens ou le client à l'impression de contacter un serveur avec une adresse publique alors que cette dernière n'est que l'adresse de la machine servant à relayer la requête. La translation de destination peut être utilisée pour des services de type équilibrage de charge sur plusieurs machines.

7.1.5 Tunnels

Les tunnels sont des mécanismes permettant de faire passer des paquets IP dans le champs de données d'autres paquets IP. Un tunnel permet de se servir d'IP comme support de niveau 2. En général cela est utilisé pour chiffrer les communications et assurer la confidentialité. En utilisant un tunnel IP une machine peut être raccordée à un réseau d'entreprise ou d'université en ayant une interface avec une adresse locale à l'entreprise (et donc les droits d'accès aux ressources) bien que physiquement connectée à un réseau internet publique avec une autre tranche d'adresses.

Chapitre 8

Réseaux et infrastructures haut débit

8.1 Utilisation des réseaux locaux

Les réseaux locaux ont des besoins qui suivent les applications qui sont utilisées. Des débits de 100 (parfois 10) Mbits/s suffisent à faire fonctionnet confortablement les services courants de type mail, web, base de données ou transfert de petits fichiers dans des réseaux locaux.

Les évolution vers les réseaux haut-débits sont la plupart du temps motivées par des applications utilisant de gros volumes de données. Ce sont maintenant des choses courantes avec l'arrivée des flux multimédia dans les réseaux locaux (échange de fichier, vidéo, son, ...). La principale utilisation du haut-débit reste tout de même (pour l'instant) la connexion de plusieurs réseaux locaux ou de lignes hauts débit de type ADSL. L'augmentation du trafic cumulé impose la mise en place de réseaux pouvant supporter des édbits très élevés pour que chaque utilisateur final puisse avoir de bonnes performances de communication.

8.1.1 Utilisation du support et haut débit

Le support physique de transmission joue un rôle essentiel dans la mise en place des protocoles de niveau 2. En effet selon la technologie utilisée les problèmes à régler à ce niveau ne seront pas les même. Les premiers protocoles (type HDLC, X25, ...) devaient se contenter de liaisons en cuivre sur de grandes distances et utilisaient des équipements rapides par rapport au débit du réseau. Le résultat était un support relativement lent ayant un BER élevé. Les protocoles devaient donc assurer des contrôles de perte de paquet réguliers dans

le réseau et souvent des mécanismes complexes de retransmission étaient présent dès le niveau 2.

L'évolution de la technologie cuivre et l'apparition de la fibre optique ont changé la donne dans les réseaux. Les paramètres ont évolés, notamment les plus importants :

- Sécurité
- Pas de sensibilité aux parasites
- Large bande passante
- Longues distances
- Erreurs : $1/25.10^{10}$ bits

Le choix des protocoles de niveau 2 doit tenir compte de la technologie utilisée :

- Support avec erreurs faible et débit élevé : correction par retransmission de bout en bout
- Support avec erreurs sensibles et débit moyen : correction par retransmission à chaque saut

8.2 Difficultés pour la gestion des protocoles

Les débits élevés imposent de revoir les protocoles qui n'ont pas été conçu avec de tels niveaux de performance en tête. Il faut en effet revoir les mécanismes de contrôle de flux et de fenêtre d'anticipation de niveau 2 pour optimiser les communications. Cela ne va pas sans une adaptation de l'architecture matérielle devant traiter de plus en plus rapidement des trames et des paquets.

Avec l'apparition des protocoles Gigabit au début des années 90, la volonté a été de réutiliser les protocoles logiciels existants pour ne pas bloquer les applications. Cela pose de nouveaux problèmes :

- Numéro de séquences trop courts : à 1 Gbit/s il faut 32 secondes pour envoyer 2^{32} octets (problème avec TCP par exemple)
- Protocole sans rejet sélectif peu efficaces (temps d'aller retour)
- Le transfert devient limité par les délais et non la bande passante

Le traitement est très vite trop lourd pour les machines, le nombre de paquets à traiter influence les performances générales (routage et réception des paquets).

Solutions de mise en place

Les réseaux rapides doivent faire évoluer les protocoles mis en place pour tenir compte de la vitesse de traitement nécessaire dans les équipements de réseau. Les équipements doivent pouvoir absor-

ber les flux et doivent éviter d'avoir une surcharge de travail en cas de perte de paquet sur une seule connexion.

- Privilégier le temps de traitement moyen par rapport à la gestion d'erreurs.
- Conserver les en-têtes les plus simples possibles
- Éviter les rétro-actions comme les mécanismes de fenêtre glissante, la fragmentation ou le démarrage lent, privilégier la réservation de bande passante.
- Optimiser le cas le plus courant, éventuellement au détriment de la gestion des erreurs (possible si on ne rencontre pas trop d'erreurs)

8.3 Haut débit dans les réseaux

Le haut débit dans les réseaux a évolué au fil du temps. La notion même de haut débit est assez variable et est passé de 10Mb/s il y a quelques années à 1Gb/s actuellement. Le terme haut débit sera sans doute réservé aux réseaux à 10Gb/s (et plus) dans les prochaines années (les réseaux à 10Gb/s sont déjà utilisés dans de nombreuses infrastructures pour interconnecter des réseaux locaux).

La suite est un tour rapide des différentes architectures de réseaux locaux qui ont été mises en place, et qui n'ont pas su résister à la mise en place d'Ethernet.

8.3.1 Quelques réseaux haut-débit

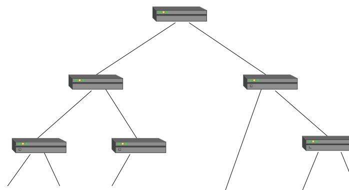
Relais de trame : Frame Relay

- Protocole de niveau 2
- Utilisé dans les liaisons longues distances
  - Circuits virtuels permanents
  - Communication point à point
  - Réservation de bande passante
  - Perte de paquets possible (CIR/EIR)
- Commutation rapide au niveau 2
- Gestion de la retransmission par les protocoles de niveau supérieurs
- ⇒ Protocole adapté à haut débit mais veillissant (40Mb/s max)

100 VG Any LAN

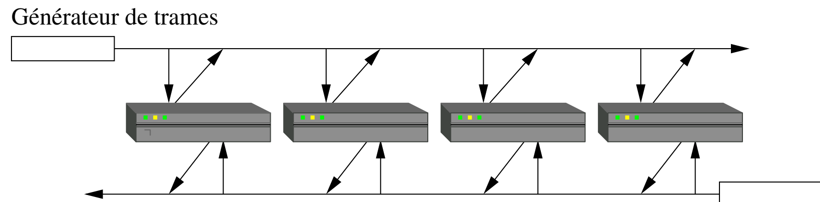
- Any LAN :
  - Compatible avec trames Ethernet et Token Ring à 100Mb/s

- MAC = DPAM (Demand Priority Access Method)
  - Pas de CSMA/CD
  - Polling : scrutation périodique de la racine
  - Accès déterministe avec 2 niveaux de priorité
- Commutateur central intelligent
- Compatible avec trames ≠ protocole ⇒ abandon de 100VG any LAN !



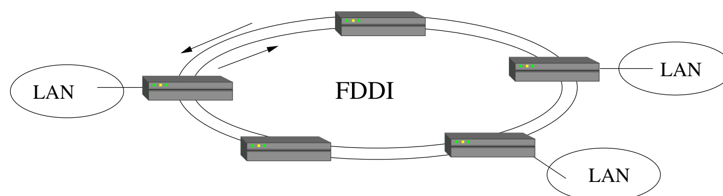
Distributed Queue Dual Bus : DQDB 802.6

- Distributed Queue Dual Bus
  - Zone géographique importante / LAN :MAN
  - Cellule de 53 octets (44 de PDU)
  - Couche d'adaptation similaire à ATM
  - Tête de réseau : émet les cellules



- Liens à 155Mb/s (similaire ATM)
- Synchronisation entre les stations
  - trames émises toutes les 125 μs
  - écouter le support pour détecter les trames
  - être capable d'émettre à un instant précis
  - le medium est coupé pour insérer l'interface
- Possibilité de réserver de la bande passante

Fiber Distributed Data Interface : FDDI



- 100 Mbit/s et évolutions Gb/s
- Objectif : interconnexion de LAN
- Architecture en double anneau
  - boucles contrarotatives
  - jeton temporisé avec priorité
- FDDI-2 = FDDI + données synchrones
  - 1 trame toutes les 125μs
- TCP/IP supporté sans problème
- Supporte jusqu'à 500 machines sur l'anneau
- Boucle de 200 km maximum (31 km de diamètre)
- LED, fibre optique multi-mode

- DQDB a été mis en avant par rapport à FDDI par l'IEEE pour les trafics synchrones (voix)
- Les extensions de FDDI pour les réseaux locaux n'ont pas résisté à l'évolution rapide d'Ethernet

ATM

- Découpage de l'information en *cellules* de 53 octets (5+48)
- Couche d'adaptation présente chez la source et le destinataire : AAL (ATM Adaptation Layer)
  - AAL 3/4 : trafic synchrone
  - AAL 5 : best effort
  - En pratique seule AAL5 est utilisée
- Architecture en étoile avec commutateur
- Mode connecté (circuits virtuels, paquets délivrés dans l'ordre)
- Pas de gestion d'erreur dans le réseau
- Débits de 155 Mbit/s à 622 Mbit/s.

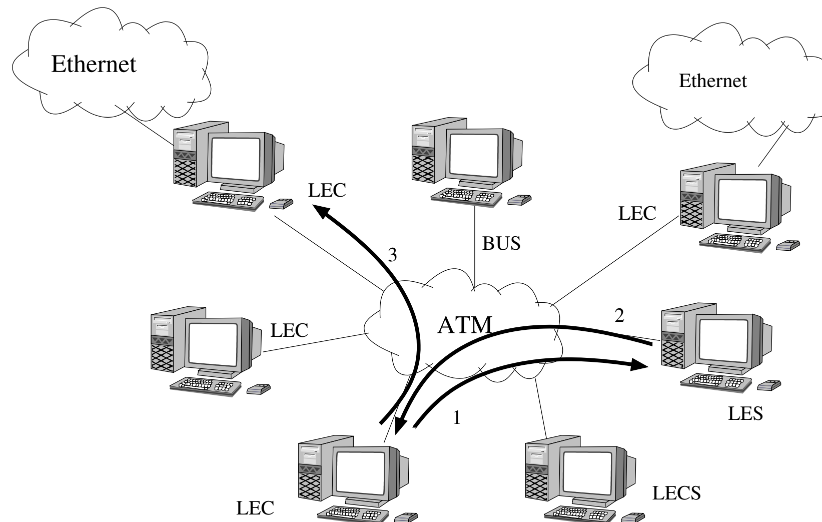
LANE sur ATM

- Interconnexion de LAN utilisant ATM
  - Cellules de 53 octets
  - Circuits virtuels permaments
- Maillage complet en circuits virtuels
- LAN émulé : émulation d'Ethernet sur ATM
  - Trames / cellules
  - Transmission non connectée / circuits virtuels
- Débits variables

Installation et configuration complexe :

- LEC : Lan Emulation Client
  - interconnection ATM/Ethernet ou ATM natif

– LES : Lan Emulation Server
  – conversion adresses IP/ATM
– BUS : Broadcast and Unknown Server
  – résolution IP/ATM par broadcast type ARP
– LECS : Lan Emulation Configuration Server
  – serveur de configuration



– Serveur connectant toutes les stations
  – Point d'engorgement
– Circuits virtuels sur toutes les connexions possibles
  – Trouver une station (LES : LAN Emulation Server)
  – Diffusion (BUS : Broadcast Unknown Server)
– Technologie trop coûteuse ⇒ abandonnée

Fiber Channel

Constat sur les communications : 2 principaux modes

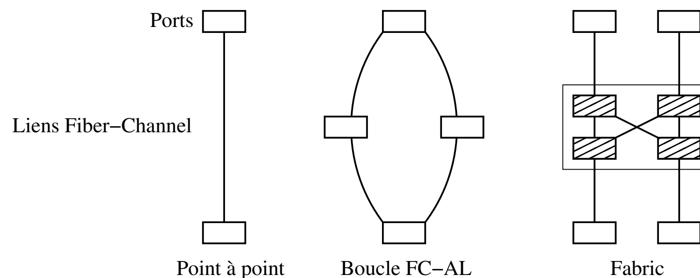
Canaux fourni une connexion point à point directe ou commutée entre les périphériques
  – Réalisés principalement en matériel, haute vitesse, peu de surcoût
  – Permettent de connecter un petit nombre de nœuds avec des adresses prédéfinies

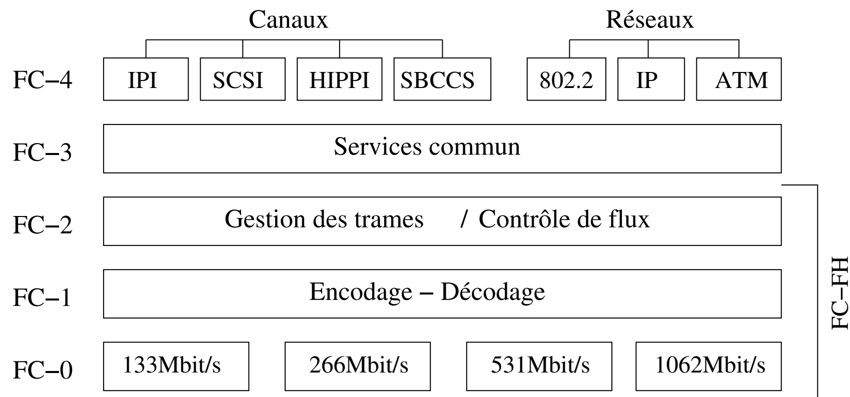
Réseaux un agrégat de nœuds distribués avec un protocole commun
  – Réalisés principalement en logiciel, surcoût de traitement important, moins rapide
  – Opèrent dans des environnements dynamiques où les communications ne sont pas anticipées

⇒ Fibre Channel tente de réunir les 2 mondes.

Fiber Channel a deux modes de communication : un premier mode pour le trafic entre 2 processeurs et un second pour le trafic entre un processeur et un périphérique. Ces modes sont appelés "channels" and "networks". Un channel est une connection directe point à point entre deux périphériques. C'est un mode dans lequel tout est fait en matériel et le transport est assuré au débit maximal avec un surcoût très faible. A l'opposé, un "network" est une agrégation de nœuds (stations de travail, serveurs de fichiers, ...) avec leur propre protocol supportant des échanges interactifs. Un network à un surcoût important en partie à cause de sa gestion logicielle et est en conséquence plus lent qu'un channel. Les networks peuvent supporter une gamme plus étendue de tâches que les channel et peuvent fonctionner dans des modes où le nombre de machine et de connections n'est pas prévu à l'avance. Les channels ne fonctionnent qu'avec un nombre limité de périphériques dont les adresses sont connues à l'avance. Fiber channel tente de réconcilier les deux modes de communication avec des interfaces pouvant combiner les utilisateurs de "channels" et de "networks".

- Réseau sur fibre optique adapté au flux de données (fichiers, video)
- Commutation de niveau 2 à 2Gb/s
- Trames de grande taille (2048 octets de charge)
- Topologies d'interconnexion variées





FC-0 lien physique, transmission série

FC-1 codage des caractères sur 10 bits, permet d'améliorer la gestion de l'horloge sur le lien série.

FC-2 contrôle de flux : bout en bout ou point à point (buffer to buffer) selon la classe de trafic. technique de retour de crédit.

FC-3 services : répartition de charge sur plusieurs liens, substitution de port, multicast

FC-4 interface entre FC et les protocoles de plus haut niveau. Permet de mélanger les types de communication.

⇒ Succès de Fiber Channel pour le transport de données dans les réseaux de stockage.

8.3.2 Ethernet et haut débit

Ethernet a été clairement le vainqueur de la compétition des réseaux rapides pour les LAN et est en passe de devenir un réseau de bout en bout. 95% du trafic transporté dans l'Internet provient déjà de machines connectées à Ethernet. Même si le nom reste le même l'Ethernet actuel n'a plus grand chose à voir avec les premiers réseaux Ethernet des année 80. La norme a évolué avec les performances pour pouvoir bénéficier de toutes les avancées technologiques. La force d'Ethernet, et ce qui en a fait le grand vainqueur de la compétition, est que cette adaptation a été faite tout en gardant une compatibilité avec les réseaux déjà installés. Un réseau Ethernet en cable coaxial à 10Mb/s peut encore entre utilisé et connecté à des réseaux utilisant du Giga Ethernet !

Fast Ethernet : fonctionnement

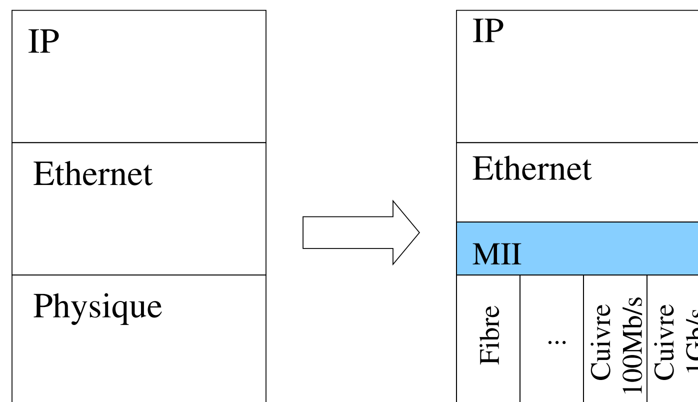
- norme 802.3u et 802.14
- Nouveau débit :
  - 100 Mb/s au lieu de 10Mb/s

  – Temps logique de 64 octets = 512 bits (inchangé)
  ⇒ Temps physique de détection d'une collision 5,12μs
  ⇒ Temps inter trames 0,96μs
- Compatible avec les réseaux Ethernet 10Mb/s
- 10/100 : auto-négociation (ou administration)

Le passage au Fast Ethernet correspond également à la généralisation de l'utilisation des commutateurs à la place des concentrateurs. Les commutateurs permettent d'avoir des liens full-duplex sur de la paire torsadée ainsi que la réduction des domaines de collisions (il n'y a pas de collision si tout est commuté).

Modifications de pile

Afin de préserver la compatibilité au niveau logiciel, une modification de la pile a été effectuée pour introduire une interface indépendante du matériel dans Ethernet. Cette partie mii (Media Independent Interface) permet de garder le protocole de détection et de résolution de collision tout en utilisant des interfaces physiques différentes.



Fast Ethernet

- Nouvelles cartes (coupleurs)
- Câble
  - Catégorie 3 (100 baseT4) : 4 paires à 25MHz, 8B/6T, duplex asymétrique
  - Catégorie 5 (100 baseTX) : 2 paires à 125MHz, 4B/5B, duplex symétrique
  - Fibre optique (100 baseFX) : 2 fibres multimodes, 4B/5B, duplex symétrique
- Le plus utilisé pour Fast Ethernet est le câble catégorie 5

Gigabit Ethernet

- Dérivé des réseaux
  - Ethernet et FastEthernet : protocole (CSMA/CD)
  - Fiber Channel : couche physique + taille de trame
- Conserve le principe CSMA/CD + Full Duplex
- Compatible Ethernet et Fast Ethernet
  - détection de collision en 5,12μs
  - ⇒ trames de plus grande taille
- Interfaces cuivre et optique

Interconnection de liens

- Interface GMII (Gigabit Media Independent Interface)
  - Extension de l'interface MII utilisée pour FastEthernet
  - Interface entre la couche MAC et la couche physique
  - Permet de connecter tout type de couche physique
  - Supporte les transferts à 10, 100 et 1000 Mbit/s
- Un mécanisme d'autonégociation permet d'adapter la vitesse de transmission selon le lien
  - Dialogue à l'établissement de la connexion physique
  - Débit : 10, 100, 1000
  - Mode half ou full duplex

Gigabit Ethernet sur cuivre

- **1000 base T**
  - norme IEEE802.3ab
  - norme importante car elle permet de d'utiliser GE dans la majorité des installations actuelles
  - Connexions half-duplex (CSMA/CD) et full-duplex
  - Câble cuivre catégorie 5 (4 paires) jusqu'à 100m
  - Un seul répéteur CSMA/CD par domaine de collision
- **1000 base CX**
  - cable paire torsadée blindée 150 Ohms
  - longueur maximale 25
  - destinées aux connexions entre serveurs

Gigabit Ethernet sur fibre

- Norme 802.3z (1000 base X)
- 1000 base SX
  - Ondes courtes (770 - 860 nm)
  - fibre multi-mode limitée à 550m en full-duplex

  – destiné aux artères de campus
– 1000 base LX
  – ondes longues (1270 - 1355 nm)
  – fibre mono-mode : 5000m full-duplex, 316m half-duplex
  – fibre multi-mode : 550m en full-duplex, 316m half-duplex
  – destiné aux artères intra-muros
– Encodage 8b/10b (signalisation à 1,25 Gb/s sur la fibre)
– Autres paramètres identiques à 1000 base T

Évolution des normes Ethernet

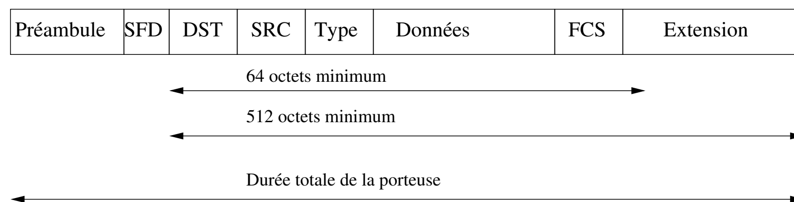
	10 base TX	100 base TX	100 base T4	100 base T2	1000 base T
Tranche de canal	51,2μs	5,12μs	5,12μs	5,12μs	5,12μs
Délai inter-trame	9,6μs	0,96μs	0,96μs	0,96μs	0,096μs
Trames	64 - 1518	64 - 1518	64 - 1518	64 - 1518	512 - 1518
Câblage	2 paires	2 paires	4 paires	2 paires	4 paires
Catégorie	Cat 3	Cat 5	Cat 3	Cat 3	Cat 5
Transmission	25 Mbaud	125 Mbaud	25Mbaud	25 Mbaud	125 Mbaud
Codage	Manchester	4b/5b	8b/6t	PAM5/5	PAM5/5

– 100 base TX
  – Une paire en émission, une paire en réception
  – 125Mbaud x 4b/5b = 100Mb/s
– 1000 base T
  – utilise 4 paires en émission et réception simultanément
  – 4 (paires) x 125Mbaud x 2bits/baud = 1Gb/s

Afin de tirer profit du passage au haut débit le protocole Ethernet a été étendu pour utiliser au mieux le débit disponible dans les réseaux Gigabits. Les deux principales innovations ont été l'extension de porteuse et le packet bursting.

Extension de porteuse

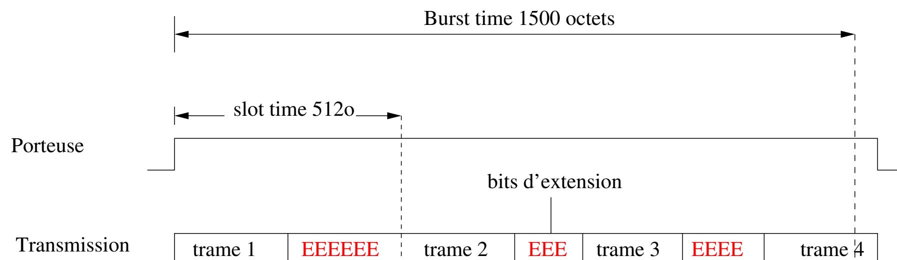
– cas des trames < 512 octets



– Extension composée des symboles spéciaux
– FCS calculé sans l'extension
– Transparent pour la couche LLC
– Problème de gaspillage de bande passante
– Performances proches de FastEthernet si le trafic est composé de petits paquets

Ethernet packet bursting

- Utilisation de l'extension de porteuse pour placer plusieurs trames



- Plusieurs trames sont envoyées jusqu'à une limite de 9000 octets
- Temps minimum inter-trame respecté
- “Jumbo frames” jusqu'à 9000 octets

Évolutions de Giga Ethernet

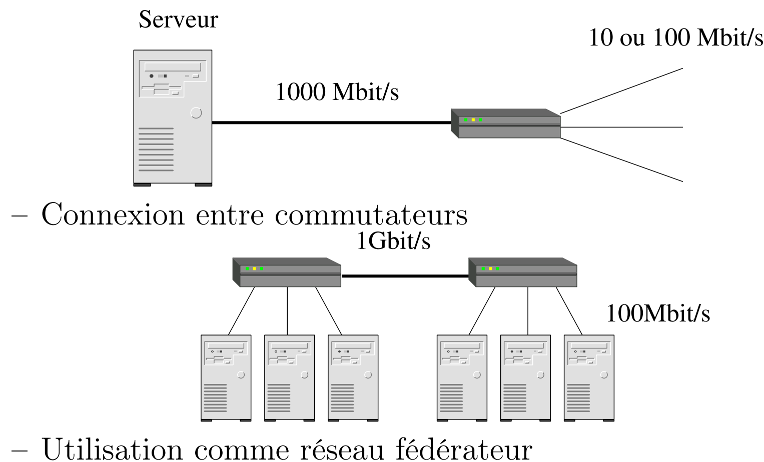
- 10 Gigabit Ethernet
  - norme 802.3ae (depuis 2002)
  - Interfaces XGMII Optiques uniquement
  - Full duplex uniquement
  - Abandon du CSMA/CD (pas de collision)
  - Contrôle du débit MAC
  - Répartition du trafic sur plusieurs liens à 2,5Gbit/s
  - Support LAN jusqu'à 40 km (Sonet, SDH, WDM)
- Les circuits 100 Gigabit Ethernet ne vont peut-être pas tarder.

Utilisation dans les LAN et MAN

- Fast Ethernet pour le lien final jusqu'aux machines
- Gigabit Ethernet pour les artères réseaux, bientôt pour les machines.
- Plus de 95% du trafic internet passe par Ethernet !
- Fiber Channel pour les échanges de données (SAN : Storage Area Network)
- Réseaux MAN
  - Évolution vers GigaEthernet et 10GE
  - LANE avec ATM : en diminution, pratiquement plus utilisé

8.3.3 Utilisation des réseaux haut débits

- Connexion serveur-commutateur

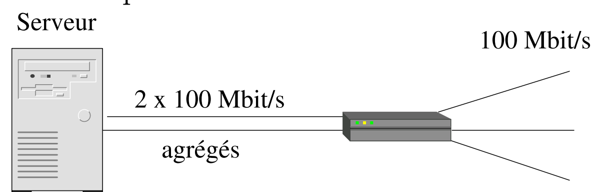


– Connexion entre commutateurs

– Utilisation comme réseau fédérateur

Aggrégation de lien

– Cumul de bande passante entre différents liens



– Couramment utilisé avec ATM et Ethernet
  – Trunking (Sun), EtherChannel (Cisco), Bonding (Linux) ...
– Permet d'avoir des liens de capacité élevée pour un moindre coût
– Solution à ne pas négliger

Problèmes importants de routage

– Commutation au niveau 2 rapide (ATM, Giga Ethernet, Fiber Channel)
  1. lire l'adresse du destinataire (6 octets)
  2. faire une comparaison dans une table pour trouver le port de sortie
  3. renvoyer la trame inchangée
– Pour autant le routage nécessite beaucoup plus de temps
  1. sortir le paquet IP de la trame
  2. en extraire l'adresse destinataire
  3. prendre la décision de routage
     – recherche dans un arbre
  4. construire une nouvelle trame

⇒ Le **routage** dans les réseaux haut débit nécessite des techniques et des matériels particuliers.

Des évolutions importantes du support physique viennent des technologies utilisées avec la fibre optique. Ces technologies sont abordées dans une autre partie du cours mais il faut retenir que sur les réseaux grande distance les paquets sont maintenant transportés sur des réseaux fibres rapides :

Sonet / SDH : Synchronous Optical Network (États Unis) / Synchronous Digital Hierarchy (reste du monde)

WDM : Wavelength Division Multiplexing : technique de communication pouvant transporter simultanément plusieurs signaux en utilisant des longueurs d'ondes différentes.

DWDM : Dense Wavelength Division Multiplexing

Aujourd'hui (à la date de rédaction de ce texte) un système DWDM est capable d'envoyer en parallèle 160 longueurs d'ondes allant jusqu'à 80Gb/s par longueur d'onde sur une seule fibre.

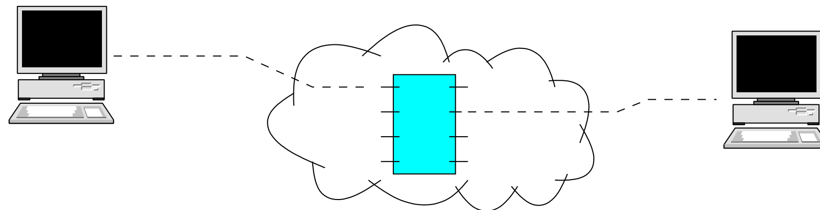
8.4 Conclusion

- Les solutions Ethernet sont de plus en plus complètes
  - Évolution vers les liens WAN avec l'ethernet 10Gb/s
  - Bientôt l'Ethernet de bout en bout
- Les réseaux haut-débit sont très sensibles au protocoles de niveaux 3 utilisés
  - Rétro-actions à éviter (fenêtres, reprises sur erreur ; ...)
  - Réservation de bande passante
  - Gestion des en-têtes efficaces
  - *Passage généralisé du routage à la commutation*

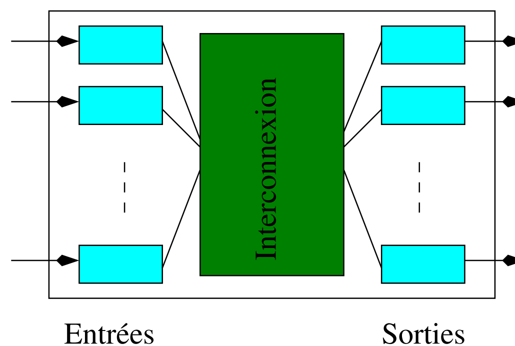
Chapitre 9

Architecture des équipements

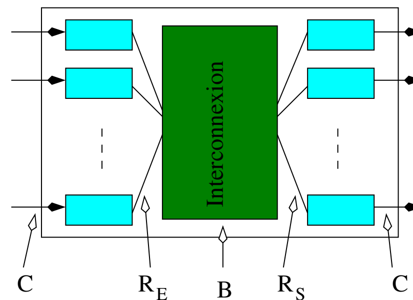
Un équipement de réseau peut-être vu comme une boite dont le rôle est d'aiguiller les paquets qui transitent sur le réseau. Cet aiguillage peut-être très simple (tout renvoyer à tout le monde : cas de l'Ethernet à sa création) mais peut aussi prendre en compte la gestion complète des paramètres du réseau (gestion de la congestion, gestion des erreurs, gestion de la qualité de service avec des priorisations de flux, filtrage du trafic, ...). Les possibilités de traitement dans les équipements de réseau sont infinies. Toutefois ces traitements ont un coût et des implications sur l'architecture matérielle des équipements.



- Ensemble de ports d'entrées/sorties
- Aiguillage entre les entrées et les sorties
- Gestion de la congestion



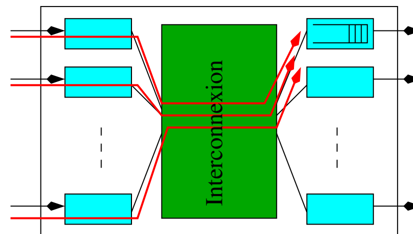
- Les E/S sont connectées par un “backplane”
  - mémoire partagée (PC, routeurs bas de gamme)
  - Bus partagé (milieu de gamme)
  - Crossbar point à point (haut de gamme)



- `C` : capacité d'entrée/sortie
- R_E : capacité d'entrée de l'interconnexion
- R_S : capacité de sortie de l'interconnexion
- `B` : débit interne de l'interconnexion
- Speedup : `B`/`C`, R_E/`C`, R_S/`C`

9.1 Fonctionnements des routeurs

File d'attente en sortie

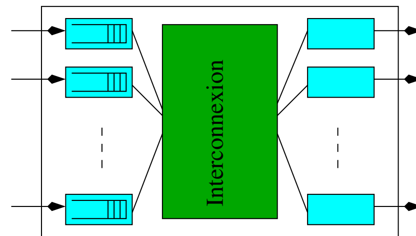


- algorithme simple
- un seul point de congestion

⇒ nécessite une accélération de sortie de N égale au nombre d'entrées.

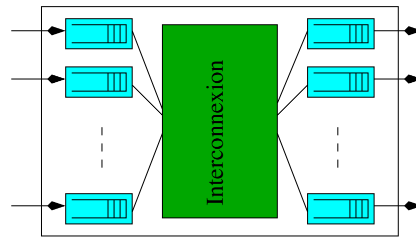
⇒ impossible à construire

File d'attente en entrée



– facile à concevoir
– stockage des paquets en entrée si contention en sortie
– possibilité d'utiliser l'algorithme de "backpressure"
⇒ efficacité limité à cause des contentions en sortie
⇒ fonctionne bien en pratique

Combinaison des solutions

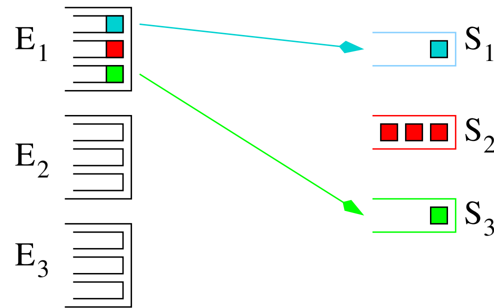


– facile à construire
– utilisation efficace avec un speedup limité
⇒ algorithmes complexes (plusieurs points de congestion)
⇒ nécessité d'avoir un contrôle de flux

Architecture des routeurs

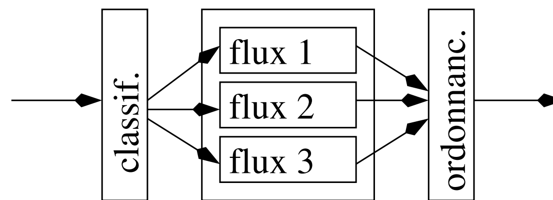
- Combinaison file en entrée, file en sortie
  - $C_E/C_S \leq 2$
- Interface d'entrée : commutation, routage, classification
- Interface de sortie : ordonnancement
- Interconnexion :
  - réseau commuté
  - $B = N$xC
- L'interconnexion permet de connecter simultanément plusieurs paires d'entrée / sortie
- Les paquets sont fragmentés en cellules de petites tailles dans l'interconnect pour éviter les contentions

Interface d'entrée



- la sortie est calculée dans l'interface d'entrée
- les interfaces maintiennent une file virtuelle par sortie

Interface de sortie



- l'interface de sortie peut lisser le trafic avec un classificateur et un ordonnanceur

⇒ voir le cours de QoS (délai, bande passante, pertes)

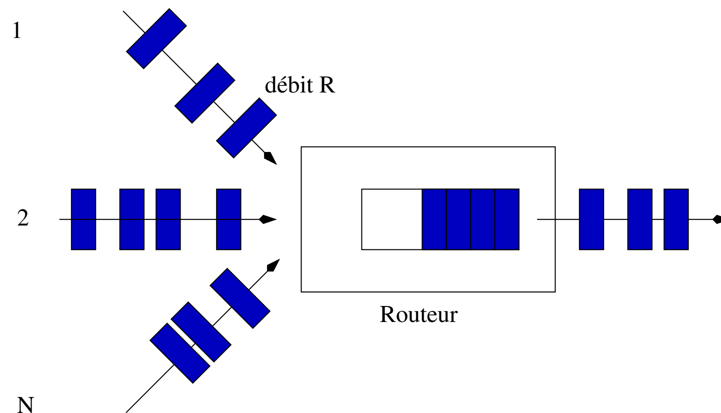
9.2 Mémorisation dans les réseaux

- Le partage de lien par multiplexage statistique est efficace
- La commutation/routage de paquets nécessite des mémoires
- Il faut éviter la perte de paquets

⇒ Utiliser de grosses mémoires

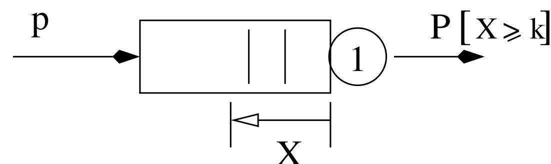
⇒ Les mémoires sont peu chères

Observations sur les performances



- Plus les buffers sont gros, moins il y a de pertes
- Si le buffer n'est jamais vide, l'interface de sortie est tout le temps active.

Théorie des files d'attente



- On peut trouver la taille d'un buffer pour un taux de perte donné
- Les pertes diminues avec l'augmentation de la taille
- Grand = mieux ? ? ?

Taille des buffers

- 1Gb de mémoire peut contenir 500k paquets et ne coûte quasiment rien

⇒ On peut faire des buffers de grande taille

- Les utilisateurs n'aiment pas les buffers
- Les opérateurs non plus
- Idem pour les concepteurs
- En fait ce n'est pas sûr que l'on en ait besoin

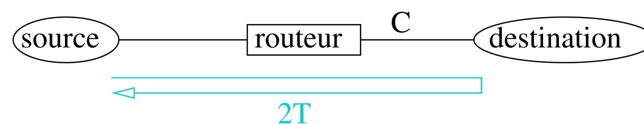
Débits importants : exemple

- réseau à 10Gb/s

  - 250ms de mémoire
  - 300Mo de mémoire
  - manipulation d'un paquet de 40 octets toutes les 32ns
- Mémoire utilisée
  - SRAM : 80 modules, 1kW, 2000$
  - DRAM : 4 modules, trop lent
- Réseaux à 40Gb/s ?

Taille de buffers

- Utilisation de TCP : contrôle de bout en bout
- Les pertes sont prévues dans le protocole
  - Fenêtre de congestion
  - algorithme du slow start
- Les pertes ne sont pas graves
- Le débit est une meilleur métrique



- Règle habituelle : $B = 2T * C$

Les routeurs gérant de nombreux flux peuvent en fait diminuer la taille des buffers. Une modification est de prendre $B = \frac{2T*C}{\sqrt{N}}$. Cette diminution peut être effectuée car les flux TCP sont désynchronisés (ils ne sont pas à leur maximum de débit en même temps) et donc le débit moyen n'est pas la somme des débits maximaux.

Avec cette nouvelle mesure on rerouve une estimation plus raisonnable : pour un réseau à 40Gb/s avec 40000 flux à 1Mb/s, la règle habituelle donne 10Gbits et la nouvelle mesure 50Mbits.

9.3 Commutation de paquets et routage

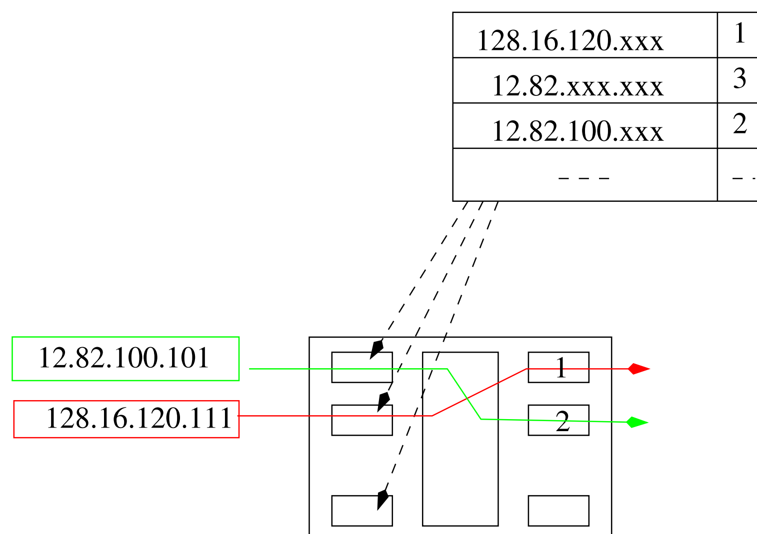
- Passage d'un bus à de la commutation de niveau 2
- Réduction des domaines de collisions
- Les ports d'entrées ont accès à une table de correspondance adresse MAC <-> port de sortie
  - Possibilité d'avoir des VLAN (filtrage dans la table : 802.1Q)
  - Possibilité de faire de la priorisation (802.1P)
- Rapidité de traitement, pas de modification de la trame
- commutateur bas de gamme 8 ports : temps de commutation de 11μs. Au maximum 90900 trames par seconde.

Niveau 3 : routage / commutation

- Les routeurs reprennent la même architecture
- Les ports d'entrées ont accès à une table de correspondance préfixes adresses IP <-> port de sortie
  - Calculée par le routage
  - Possibilité d'avoir des VLAN par adresse IP
  - Possibilité d'avoir de la priorisation
- Commutation de niveau 3 : modification de la trame à la volée
  - Les réseaux d'entrée et de sortie doivent être de même nature
  - Adresses MAC (source et destination)
  - Recalcul du CRC (très long)
- Commutation de niveau 3 : reconstruction de trame
  - Cas pour lequel les réseaux de niveau 2 sont différents

Problème du routage

- Identifier l'interface de destination en fonction de l'adresse destination IP
- Les tables de commutations ne stockent que la correspondance entre préfixes d'adresses IP et port de sortie
- Le choix de la route doit donc trouver le port de sortie ayant le préfixe de correspondance le plus long

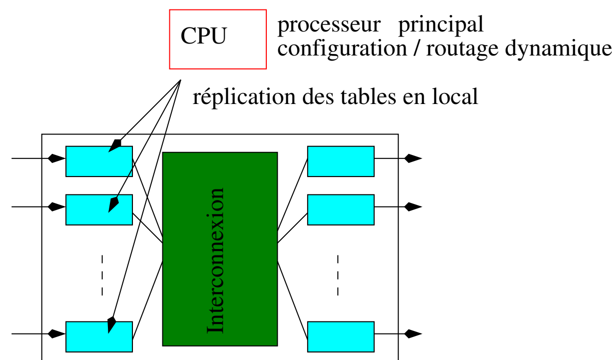


- La structure de données utilisée pour représentée la table de correspondance est crutiale
- Une implémentation naïve fonctionne en $O(m)$ où m est la longueur du préfixe (32 pour IP)

? Quelle est cette représentation ?

– minimiser le nombre de lectures dans la table
– minimiser la taille de la structure
⇒ Utiliser une structure hiérarchique
– Premier niveau : couvre tous les préfixes sur 16 bits
  – Utilisation d'un bit par préfixe (présent / non présent)
  – Mémoire utilisée $2^{16} = 64Kb = 8Ko$
  – On stocke un pointeur vers le niveau suivant si le préfixe est valide (densité)
– Deuxième niveau, troisième niveau ...
– Les structures sont optimisés pour être rapides en lecture
– Le construction de la structure est plus délicate
  – Les tables de routages ne changent pas souvent

Architectures



– Le processeur principal traite les paquets à destination du routeur
  – configuration - maintenance
  – services dynamique (routage)
– La table de routage est recopiée dans chaque port d'entrée pour avoir un accès rapide et parallèle (pas de mémoire partagée)
– Les ports sont autonomes
– Lorsqu'un port reçoit un paquet à destination du routeur il l'envoit au processeur principal qui recalcule les tables et les redistribue
– Construction de processeurs spécialisés dans le traitement de paquets
– Network Processors
  – Intel IXP
  – IBM
  – Motorola
  – ...

Exemple de routeurs haut de gamme

- BlackDiamond 6800
  - jusqu'à 1400 ports 100Mb/s
  - interconnexion de commutation 768Gb/s
  - 192 MPPS (Million de paquets par secondes)
  - interventions niveaux 2,3,4 et applicatif
- BlackDiamond 12000
  - 48 ports 10Gb/s Ethernet ou 480 ports Gb/s
  - 1.6Tb/s en commutation
  - 1,2 millions de routes IPv4/IPv6
  - ...

9.4 Conclusion

- Plus on remonte dans les niveaux, plus le traitement à fournir dans l'interface d'entrée est important
- Les performances des routeurs varient grandement si l'on active des services de haut niveau
  - filtrage d'adresses IP
  - filtrage de port UDP ou TCP
  - filtrage de contenu de requête
  - ...
- Unité de mesure : MPPS Millions de paquets par secondes

Chapitre 10

Normalisation et standardisation

10.1 Standardisation et modélisation des réseaux

La modélisation des réseaux permet de comparer les différents protocoles entre eux en plaçant des points de repère et une séparation claire des différentes tâches présentent lors d'une communication. Cette séparation en tâche, appelées couches, permet également d'obtenir une modularité importante. C'est cette modularité, i.e. la capacité d'organisation et d'interaction entre des logiciels différents mais devant remplir une tâche équivalente, qui a permis le développement rapide des réseaux et leur inter-opérabilité.

Les comparaisons entre niveaux sont en général de deux ordres :

- Comparaison de niveau d'abstraction
- Comparaison fonctionnelle

Il existe deux principaux modèles utilisés dans les réseaux modernes :

- Le modèle OSI : *Open Systems Interconnection Basic Reference Model.*
- Le modèle Internet TCP/IP illustré dans les précédents chapitres.

Le modèle OSI a été proposé en 1977 par l'*International Organization for Standardization* (ISO, http ://www.iso.org/). Dans ce modèle, un système en réseau utilise un pile constituée de 7 couches. Chacune des couches est responsable d'une partie des fonctionnalités jugées nécessaires au bon fonctionnement de l'ensemble. Les couches interagissent entre elles de façon verticale : une couche n pilote les services d'une couche $n-1$ et propose des services de plus haut niveau à la couche $n+1$. Les 7 niveaux de la pile OSI sont représentés

sur la figure 10.1.

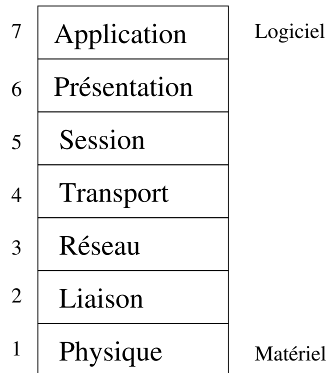


Figure 10.1 – Pile OSI de référence

10.2 Le modèle OSI

Le modèle OSI a été créé pour la standardisation des divers protocoles employés dans les réseaux d'ordinateurs "ouverts". Ce modèle d'organisation est un modèle de référence “papier” et ne correspond pas réellement à une technologie réseau “réelle”. Il permet cependant de placer les principes de fonctionnement de manière indépendante d'une implantation.

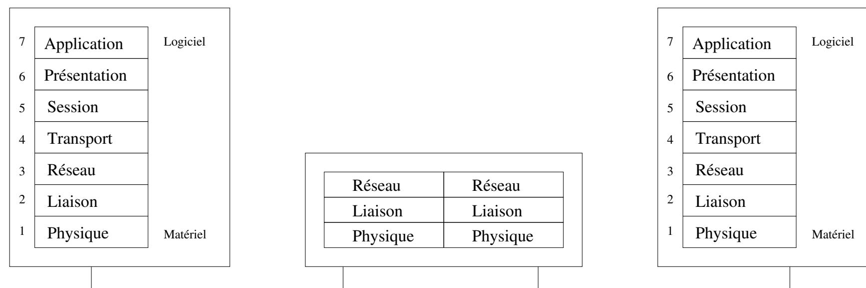


Figure 10.2 – Pile OSI de référence

La figure 10.2 représente les différentes couches utilisées lors de la communication entre deux machines interconnectées par un routeur. On peut y reconnaître facilement les niveaux utilisés dans le modèle Internet ainsi que ceux n'apparaissant pas explicitement dans les différents protocoles utilisant IP.

1. La couche physique

– Responsable de :

  - Transmettre des séquences de bits via un médium de communication
- Verrous :
  - Interface mécanique / électrique
  - Temps / bit
  - Distance

2. La couche liaison de donnée

- Offrir un lien de communication sans erreur
- Gestion des accès multiples
- Gestion des destinataires sur le lien
- Verrous :
  - Utilisation efficace du support
  - Partage équitable du temps de parole
  - Construction des trames et adressage local

3. La couche réseau

- Routage
  - Sélectionner les itinéraires
- Fragmentation & réassemblage
- Translation entre réseaux de différents types
- Verrous :
  - Etre indépendant du matériel (adresse et taille des paquets)
  - Interconnecter tous les types de réseaux possibles

4. La couche transport

- Offrir un lien virtuel entre des processus terminaux
- Contrôle de flux de bout en bout

- Verrous :
  - Communication fiable
  - Détection d'erreur
  - Abstraction du monde paquet utilisé jusqu'à la couche 3

5. La couche session

- Localiser les services
- Établir, gérer et terminer les sessions entre applications
- Passage d'un mode paquet à un mode de transfert bidirectionnel sur plusieurs échanges

6. La couche présentation

- Cryptage des données
- Compression des données
- Conversion des données

- Beaucoup de protocoles ne possèdent aucune couche présentation (qui est donc dans l'application)

7. La couche application

- Tout ce qui n'est pas pris en compte par les autres couches !
- Protocoles de niveau applicatif

10.3 Le modèle Internet TCP/IP

Modèle plus simple (en nombre de couches) que le modèle OSI.
Modèle de référence pour les applications sur Internet.

- Chaque couche doit ajouter du contrôle d'information sur les données pour faire son travail
- Information le plus souvent accolées au données avant de les donner à la couche inférieure
- Quand les données & les informations de contrôle sont délivrées, la couche à l'autre extrémité terminal utilise les informations de contrôle.

Chapitre 11

Bibliographie et liens

Bibliographie

[1] The Internet Engineering Task Force. en ligne, http ://www.ietf.org/, June 2006.

[2] James F. Kurose and Keith W. Ross. *Computer Networking A Top-Down Approach Featuring the Internet.* Addison Wesley Publishing Company, 2000. ISBN 0201477114.

[3] Richard Stevens. *UNIX Network Programming.* Prentice Hall, 1990. ISBN 0-13-949876-1.

[4] Richard Stevens. *TCP/IP Illustrated, Volume 1 : The Protocols.* Prentice Hall, 1994. ISBN 0-201-63346-9.

[5] Richard Stevens. *TCP/IP Illustrated, Volume 2 : The Implementation.* Prentice Hall, 1995. ISBN 0-201-63354-X.

[6] Richard Stevens. *TCP/IP Illustrated, Volume 3 : TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols.* Prentice Hall, 1996. ISBN 0-201-63495-3.

[7] Richard Stevens. *UNIX Network Programming, Volume 1, Second Edition, Networking APIs : Sockets and XTI.* Prentice Hall, 1998. ISBN 0-13-490012-X.

[8] Richard Stevens. *UNIX Network Programming, Volume 2, Second Edition, Interprocess Communications.* Prentice Hall, 1999. ISBN 0-13-081081-9.

Ressources et liens utiles

- Site web du département : Moodle
  http ://moodle.insa-lyon.fr/course/view.php ?id=467

- Renater, Le Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche :
  http ://www.renater.fr/
- Rocad, plan du réseau :
  http ://cisr.univ-lyon1.fr/reseau/indexplan.html

- IETF, The Internet Engineering Task Force :
  http ://www.ietf.org/
- RFC : http ://www.ietf.org/rfc.html

- Site web du livre "Analyse structurée des réseaux" de J. Kurose et K. Ross
  - le livre : `http://wps.aw.com/aw_kurose_network_2/`
  - les applet Java : `http://wps.aw.com/aw_kurose_network_2/0,7240,227093-,00.html`

Pour aller plus loin / les prochains modules

3TC-PRS proposé au second semestre, permet de mettre en pratique les notions vues dans les cours de système d'exploitation (SDE) et d'algorithmique et programmation (AGP) sous forme d'un projet de programmation.
http ://moodle.insa-lyon.fr/course/view.php ?id=468,

4TC-ASP Les interconnexions de réseaux, les réseaux de grande taille, ainsi que les problématiques de routage dynamique associées à ces réseaux de réseaux sont présentées dans l'année de 4TC dans le cours ASP.
http ://moodle.insa-lyon.fr/course/view.php ?id=481.

Annexe A

Travaux dirigés

NET– TD 1 : Réseau physique et Ethernet

1 Questions préliminaires :

- Expliquer l'utilité d'une couche MAC
- Qu'est-ce qu'une adresse MAC ?
- Comment est-elle attribuée ?
- A quoi sert-elle ?

2 Petits exercices :

Soit un réseau local Ethernet (IEEE 802.3) à 10 Mbit/s constitué d'un bus de 500m sur lequel les signaux se propagent à 200m/μs.

- Quel est T le temps A/R sur le support ?
- Quel est le temps d'occupation du support pour la taille minimum d'une trame ?
- Quel est alors le débit utile ?

Soit un réseau Ethernet à 10 Mb/s grâce auquel 4 stations (S1, S2, S3 et S4) communiquent. On considère que T, le temps A/R sur le support, est égal à 5μs (on rappelle qu'au-delà de la valeur Tmax=51,2μs, on ne peut plus utiliser Ethernet).

Pour simplifier, on considère que tous les messages échangés ont une longueur constante et égale à 1000 octets. On s'intéresse aux trames sur le bus à partir de l'instant t0 (avant t0, on suppose que les stations n'ont pas émis). A t0, S3 a un message à émettre. A t0+2μs, S4 a un message à émettre.

- Que se passe-t-il ? Pourquoi ?
- Que se passe-t-il dans Ethernet lorsqu'une collision se produit ?
- Donner l'organigramme d'émission d'une trame.

3 Étude de cas : câblage d'un bâtiment

On considère l'aménagement du bâtiment Claude Chappe « Télécommunication Services et Usages ». On occupe les 3 étages du bâtiment. Tous les ordinateurs sont reliés par Ethernet 100Mbits/s (principalement pour l'échange de données mais aussi pour la téléphonie sur IP).

Câblage

- Hauteur d'un étage : 3m
- Longueur et largeur : 37m x 17m pour TC
- Longueur de câblage : deux fois la distance « à vol d'oiseau » (les câbles ne peuvent être tirés qu'à angle droit).

Trafic

- Taille moyenne des trames MAC échangées : 1000 octets (avec l'entête)
- En période de pointe, chaque station gère 500 trames par seconde à transmettre.

En comptant le nombre de bureau et le nombre de salles de TD/TP qui sont équipées on peut compter un ensemble d'à peu près 500 prises pour TC et 300 pour le CITI. Les machines dans le bâtiment sont réparties assez simplement (15 pour l'admin, 24 pour la salle réseau, 15 salle Radiocom, 15 par salle de TD, un total de 30 bornes wifi).

- Proposer un câblage des étages. Donner une topologie générale en représentant les différents équipements. Quelles sont à votre avis les contraintes les motivations pour la mise en place du câblage d'un bâtiment ?

Les équipements qui sont disponibles sont des commutateurs 24 ports. Ces commutateurs sont situés dans une pièce au $2^{\text{ème}}$ étage. La salle contient également les serveurs de machines virtuelles, les serveurs de disque et la sortie du bâtiment.

- Estimez les différents débits et le nombre de paquets par seconde qui transitent sur les différents liens de l'architecture.
- Sur quels liens peuvent se situer les problèmes ?
- Donnez les ordres de grandeurs en nombre de paquet commutés par seconde.

NET– TD 2 : Ethernet, IP et routage

1 Désassemblage de trames Ethernet

Un analyseur de réseau est connecté sur un LAN Ethernet et permet d'extraire le contenu des trames circulant sur celui-ci (le préambule, le SFD ainsi que le CRC et les octets de bourrage sont supprimés). Une capture du réseau donne les résultats suivants :

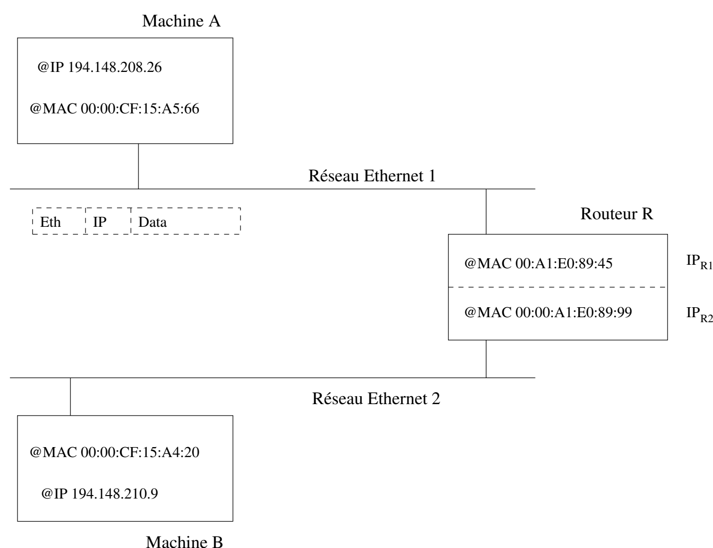
```
FF FF FF FF FF FF 08 00 20 02 45 9E 08 06 00 01 08 00 06 04 00
01 08 00 20 02 45 9E 81 68 FE 06 00 00 00 00 00 00 81 68 FE 05
08 00 20 02 45 9E 08 00 20 07 0B 94 08 06 00 01 08 00 06 04 00
02 08 00 20 07 0B 94 81 68 FE 05 08 00 20 02 45 9E 81 68 FE 06
```

Questions :

1. Désassemblez les octets précédents en vous aidant des formats de paquets Ethernet et ARP.
2. Quelles sont les adresses MAC et IP des machines concernées ?

2 IP : Encapsulation, ARP

On considère le réseau suivant composé de deux sous-réseaux Ethernet (1 et 2) raccordés par un routeur IP.



Questions :

1. Sur le réseau Ethernet 1, `A` envoie des données à la passerelle `R`. Représentez les trames successives échangées entre `A` et `R`.

2. L'hôte `A` désire envoyer un paquet à l'hôte `B`. Complétez le schéma ci-dessus pour faire figurer les adresses sources et destinations contenues dans la trame Ethernet et dans le paquet IP.
3. Quelle est la taille minimale des masques des réseaux R1 et R2 pour que le routage soit bien mis en route ?

3 Adresses Internet et subnetting

1. On désire diviser un réseau possédant le préfixe 129.178/16 en 60 sous-réseaux. Combien de machines au maximum pourra-t'on connecter sur chaque sous-réseau ? Quel sera le masque de sous-réseau ?
2. Considérons un réseau utilisant un masque égal à 255.255.248.0. Les trois stations d'adresses respectives : 194.148.208.26, 194.148.216.145 et 194.148.210.32 appartiennent-elles au même réseau ? Quelle est la plage d'adresses utilisée ? Définir l'adresse de diffusion sur le réseau local.

4 Routage et Adressage

Une société dispose du réseau suivant sur son site. Son fournisseur d'accès Internet (FAI) lui fournit le préfixe 192.108.116.0/22.

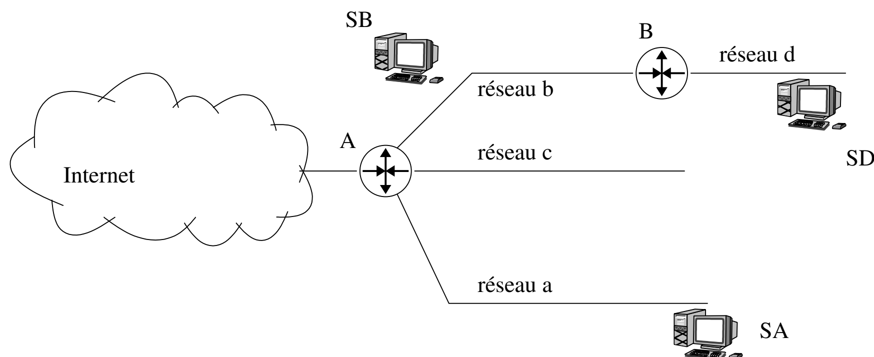


Figure A.1 – Réseau IP avec routage

1. Donnez l'intervalle des adresses possibles pour cette société ;
2. Combien de bits sont nécessaires pour numéroter les sous-réseaux ?
3. On suppose qu'il n 'y aura jamais plus de 60 stations par sous-réseau. Proposer un plan de numérotation pour l'entreprise, i.e., pour chaque réseau a, b, c et d donnez le numéro de sous-réseau, et la plage d'adresses des machines dans ce sous-réseau.

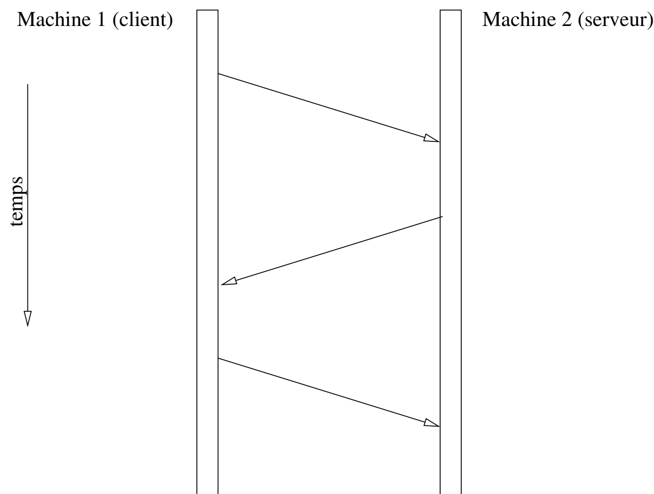
L'ingénieur système décide de numéroter les équipements dans l'ordre croissant en commençant par la première adresse IP disponible sur le sous réseau et les interfaces des routeurs dans l'ordre décroissant en partant de la dernière adresse IP disponible sur le sous réseau.

4. Donnez une adresse aux routeurs sur le schéma précédent.
5. Donnez la table de routage des routeurs A et B
6. Quelle configuration donnez vous pour les machine SA, SB et SD ?

NET– TD 3 : IP et TCP

1 Gestion de connexion

Le diagramme suivant reproduit le mécanisme d'établissement de connexion de TCP.



L'application qui demande la connexion envoie un segment TCP avec le bit SYN à 1 et le bit ACK à 0 pour initier la connexion. Le paquet envoyé indique également la taille de segment que la machine est prête à recevoir (initialisation de la fenêtre d'émission TCP du serveur) ainsi que le numéro de port de communication pour l'application cliente et le numéro de port de connexion pour atteindre le serveur.

Si aucune application n'a ouvert le port destination demandé la machine 2 renvoie un paquet avec le bit RST à 1 pour signifier le refus de connexion. Dans le cas favorable, le segment est passé à l'application qui choisit ou non d'autoriser la connexion. En cas d'autorisation le segment renvoyé comporte le bit SYN à 1, le bit ACK à 1 ainsi que la taille de segment accepté et les bon numéros d'acquittement.

Le troisième paquet est renvoyé par le client et sert à confirmer la mise en place de la connexion.

Question :

1. Représentez les informations échangées dans les segments ainsi que les entêtes.
2. Quelles peuvent être les valeurs des numéros de séquence et d'acquittement dans les 3 premiers paquets échangés ?

3. Une connexion TCP est identifiée de façon unique par un quadruplet
   <IP source, port source, IP destination, port destination>
   Que se passe t'il lorsqu'une deuxième machine ou une deuxième application demande une connexion au serveur ?
4. Sachant que les communications TCP sont bidirectionnelles, proposez un mécanisme de fin de connexion pour TCP.
5. A la vue des mécanismes d'établissement de connexion et de fermeture, quels sont les états possibles d'une connexion TCP (du point du vue du client et de celui du serveur) ?

2 Politique de transmission

Les transmissions de données se font dans TCP en utilisant une fenêtre glissante pour temporiser et grouper les octets à envoyer dans le réseau. Si l'on ne fait pas attention, ce mécanisme peut s'avérer totalement inefficace.

Considérons son fonctionnement normal : le destinataire annonce dans les acquittements la taille de données qu'il est prêt à recevoir (place libre dans la fenêtre). Ces données sont ensuite lues par l'application à un rythme/débit différent de celui du flux TCP entre les machines. Par exemple, si une machine possède une fenêtre de 4Ko et qu'elle reçoit un paquet de 2Ko, la taille renvoyée dans le segment d'acquittement pour la fenêtre sera de 2Ko. Si le prochain segment TCP arrive sans que l'application n'ait retiré de données alors le prochain acquittement aura une taille de fenêtre 0 (nulle).

L'émetteur doit alors attendre une nouvelle annonce de la part du récepteur pour recommencer à émettre.

2.1 Algorithme de Nagle

Cet algorithme est une première modification du mécanisme envoi/acquittement pour les cas où l'émetteur envoie les données octet par octet dans le flux TCP. Ceci est particulièrement le cas dans les sessions interactives type Telnet/Ssh pour lesquelles les informations envoyées sont les caractères tapés au clavier.

Questions :

1. Quel est le volume échangé sur le réseau pour chaque octet transféré ?
2. Proposez une solution pour améliorer le débit sans pour autant augmenter de façon significative la latence d'envoi des octets.

2.2 Algorithme de Clark (silly window)

Cet algorithme correspond au cas où l'émetteur envoie des segments de grande taille mais où l'application ne lit les données qu'octet par octet sur la machine destinataire.

Questions :

1. En reprenant le mode de fonctionnement normal, quel est le problème posé dans la gestion de la fenêtre ?
2. Proposez une solution pour améliorer le comportement de TCP lorsque le destinataire retire les données octet par octet.

Remarque sur la politique de transmission et le contrôle de congestion
Le contrôle de congestion est assuré dans TCP par l'algorithme de démarrage lent (slow start) de Jacobson. Cet algorithme permet de savoir quelle est la taille de segment maximum qui peut être envoyé dans le réseau sans perte. Ceci est différent de la gestion de la fenêtre que l'on vient de voir (taille de buffer disponible dans une machine). Lorsque TCP choisit de créer un segment il doit prendre la taille minimum entre les deux valeurs (fenêtre annoncée et taille max de segment) pour être sûr de ne pas déborder la capacité de réception du destinataire (taille de la fenêtre) et de ne pas envoyer des paquets trop gros dans le réseau (taille de segment maximum).

NET– TD 4 : TCP

1 Questions diverses

1. Soit une connection TCP entre un serveur A et un serveur B. Supposons que les segments TCP circulant de l'un à l'autre soient dotés d'un numéros de port d'origine x et d'un numéro de port de destination y. Quels sont les numéros de port d'origine et de destinations des segments circulant de B vers A ?
2. Pour quelles raisons un développeur pourrait préférer le protocole de transport UDP au protocole TCP pour ses applications ?
3. Une application peut-elle bénéficier d'un service de transfert de données fiable si elle repose sur UDP ? Si oui, comment ?

2 Contrôle de congestion dans TCP

Dans cette exercice nous allons tenter de trouver une approximation du débit de TCP employant le mécanisme AIMD (*Additive Increase Multiplicative Decrease*).

La figure A.2 montre l'évolution en *dent de scie* de la taille de la fenêtre de l'émetteur en fonction du temps. W est la taille atteinte maximale de la fenêtre (mesurée en paquets). Dans cette question, vous pouvez supposer que tous les paquets sont de longueur P bits et qu'exactement un paquet est jeté chaque fois que la taille de la fenêtre atteint la valeur W.

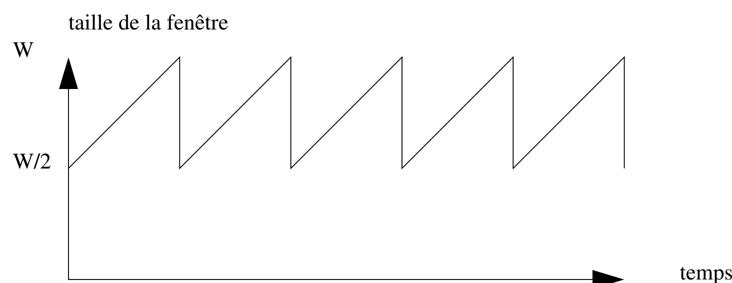


FIGURE A.2 – Evolution en *dent de scie* de la taille de la fenêtre de l'émetteur en fonction du temps.

1. Pourquoi TCP emploie un accroissement additif (*Additive Increase*) plutôt que multiplicatif ?

2. Si on ignore la phase de *slow start* en début de flux, montrez que l'émetteur envoie ses paquets à un débit moyen donné par :

$$\bar{R} = \frac{3}{4}\frac{W}{\mathrm{RTT}} \text{ paquets par seconde} \tag{A.1}$$

   où RTT est le temps d'aller-retour (*Round-Trip-Time*) qui sera supposé être constant.

3. Montrez que la fraction L de paquets jetés est donnée par l'expression suivante :

$$L = \frac{1}{\frac{3}{4}W\left(1+\frac{W}{2}\right)} \tag{A.2}$$

   On rappelle l'hypothèse faite en début d'exercice qui suppose qu'exactement un paquet est jeté chaque fois que la taille de la fenêtre atteint W.

4. À partir de vos réponses aux questions précédentes, et en supposant que la taille maximale de la fenêtre W est très grande, montrez que le débit moyen de l'émetteur est donné par :

$$\bar{R} \approx \frac{\frac{3}{4}\sqrt{\frac{8}{3}}}{\sqrt{L}\cdot\mathrm{RTT}}P \approx \frac{1.22P}{\sqrt{L}\cdot\mathrm{RTT}} \tag{A.3}$$

   On pourra supposer que $1+\frac{W}{2} \approx \frac{W}{2}$ pour W grand.

5. Le débit utile (*goodput*) d'une connexion TCP est définit par le débit des données utilisateur (application) envoyées la première fois, *i.e.*, le débit utile ne prend pas en compte les données qui sont retransmises. Le débit utile sera-t-il plus grand ou plus petit que $\bar{R}$?

6. On suppose que la totalité des donnée d'une fenêtre est retransmise chaque fois d'un paquet est jeté. Réécrire l'équation de la question 4 pour montrer le débit utile du flot.

NET– TD 5 : Utlisation et paramètres du serveur web Apache

A. Fraboulet, d'après un sujet de S. Frénot et F. Le Mouël

Le World Wide Web repose essentiellement sur deux normes : `HTML` (HyperText Markup Language) définissant le format de fichiers hypertextes et `HTTP` (HyperText Transfer Protocol) définissant les communication entre un client (navigateur) et un serveur.

Le rôle du serveur Web est :

1. accepter les requêtes des clients ;
2. rechercher la ressource ;
3. préparer l'emballage de la ressource demandée ;
4. insérer la ressource à la suite de l'emballage et renvoyer le tout au client.

1 Quelques mots sur le protocole HTTP

Une ressource peut se composer d'un ensemble de fichiers ou d'un exécutable accessible par un serveur Web. Une ressource est identifiée par son type MIME (MultiPurpose Internet Mail Exchange).

Requêtes HTTP : Les serveurs Web utilisent le protocole HTTP, le protocole définit le format des requêtes ainsi que le format des réponses. Les types de requêtes possibles sont :

- `GET` pour demander une ressource
- `POST` pour transmettre de l'information à un serveur
- `HEAD` pour demander des entêtes de ressources
- `PUT` pour déposer une ressource

Une requête HTTP envoyée par le client est constituée de 3 parties : une ligne de requête, des en-têtes et un corps. Le format d'une requête du client vers le serveur est :

```
<Type de Requête> Nom_de_la_ressource Version_protocole
[paramètres_client]
<saut de ligne>
```

Voici l'exemple donné pour un GET :

```
GET / HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; U; Linux i686; fr-FR; rv:1.8.1.17)  [...]
Accept: text/xml,application/xml,application/xhtml+xml,text/html; [...]
Accept-Language: fr,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
                              <---- Ligne vide obligatoire (CR LF seuls)
```

Une ligne de requête est composée de 3 éléments séparés par des espaces : la méthode (ici GET, la ressource sous forme de chemin absolu et la mention de version HTTP (HTTP/1.0). Cette ligne ainsi que la ligne vide suivante sont les seules obligatoires. La version 1.1 du protocole HTTP impose d'ajouter la ligne Host : afin d'indiquer quel est le nom public démandé.

Réponse HTTP La réponse se décompose en deux parties : un entête, qui identifie le type de la ressource renvoyée, une ligne vide et la ressource elle même. La ligne vide permet d'identifier la fin de l'entête. Il est possible que le serveur renvoie plus d'information dans l'entête http (durée de vie du document, type de serveur ...). Une réponse type prend la forme donnée ci dessous :

```
HTTP/1.1 200 OK
Server: Apache/2.2.3
Content-Type: text/html; charset=iso-8859-1
                              <---- Ligne vide obligatoire (CR LF seuls)
<html>
<body>
Bonjour à tous
</body>
</html>
```

La ligne d'état (la première), seule obligatoire, est composée de trois éléments séparés par des espaces un identificateur de version HTTP (`HTTP/1.1` dans notre cas), suivi d'un code de diagnostic, lui-même suivi d'un message de diagnostic correspondant. En cas d'erreur, le corps de réponse doit tout de même être constitué d'un document HTML valide expliquant l'erreur.

Pour simuler une connexion cliente sur un serveur Web, on peut faire un telnet sur le port de connexion :

```
telnet machine-cible port-tcp
```

On dialogue alors selon le protocole `HTTP` sur le port 80 de la machine de destination.

Exercice : Comment simuler la récupération de la ressource index.html sur telecom.insa-lyon.fr ? Que ce passe t'il lorsque l'on clique sur un hyperLien ?

2 Configuration d'un serveur web

Afin de tester la suite, réalisez une page Web simple pour en faire votre page de test.

Cette partie a pour objectif d'installer un serveur Web personnel indépendamment d'un accès administrateur.

Vous devez télécharger les fichiers de configuration depuis Moodle pour installer un serveur Web (il ne s'agit que des fichiers de configuration et non pas de l'exécutable) sur votre compte utilisateur.

Éditez les fichiers
`httpd/conf/http2.conf`
`httpd/conf/commonhttpd.conf`
afin d'adapter les paramètres donnés au début des fichiers, notamment le paramètre `ServerRoot`. N'hésitez pas également à jeter œil sur les autres paramètres.

2.1 Démarrer, tester et arrêter le serveur

Lancer le serveur avec la commande

```
/usr/sbin/apache2 -f [chemin]/httpd/conf/httpd2.conf
```

Vous pouvez observer les fichiers de log avec la commande

```
 $ tail -f <fichier>
```

Quelles sont les traces générées dans les fichiers de log ?

Vous pouvez maintenant atteindre votre serveur web avec un navigateur en le faisant pointer sur
http ://localhost :8000/.

Exercice : A l'aide de la commande `telnet`, simulez une requête HTTP sur le serveur pour retrouver la page par défaut et observez les fichiers de logs du serveur.

Pour contrôler votre serveur

- Pour savoir si votre serveur tourner : `ps ax | grep apache`
- Pour arrêter votre serveur : `killall -SIGTERM apache2`
- En utilisant le fichier httpd.pid :
  `kill -SIGTERM ‘cat $HOME/httpd/lock/httpd.pid‘`

3 Indirections et pages des utilisateurs

Apache permet de configurer des répertoires pour accueillir les pages des utilisateurs d'une machine. Les adresses deviennent

```
http://machine:port/~utilisateur/index.html
```

Le répertoire dans lequel les fichiers doivent se trouver pour qu'un utilisateur puisse déposer sa page web est `$HOME/public_html`.

La serveur web fait donc de manière implicite la redirection dans des répertoires différents selon les ressources demandées.

Exercice : Modifiez les fichiers de configuration (si besoin) et créez votre répertoire public_html pour mettre en place une page correspondant à votre nom d'utilisateur.

Ce type de configuration est une première source de redirection pour les adresses web. De nombreuses autres possibilités sont réalisables. On peut ainsi rediriger de façon transparente vers d'autres sites web, changer les noms des fichiers renvoyés ou bien encore créer le contenu à la volée comme dans la prochaine section.

4 Pages dynamiques : CGI / PHP / Java

L'interface CGI permet au serveur de donner la main à un exécutable chargé de renvoyer la ressource au client. L'exécution de ce programme génère donc le contenu complet (y compris l'entête) qui sera renvoyé au client. Du point de vue du client il n'y a pas de différence avec un fichier qui aurait été fabriqué à l'avance et stocké sur le serveur web.

Pour programmer un script CGI n'importe quel langage sachant récupérer une entrée standard et sachant écrire sur une sortie standard peut être utilisé. Certains langages sont cependant plus adaptés que d'autres (comme PHP, Perl ...).

Un exemple de script shell pourrait ressembler à celui-ci :

```
#! /bin/sh
echo "Content-Type: text/html"
echo "" # ligne vide
echo "<html><body> "
echo bonjour $REMOTE_ADDR
echo "</body></html>"
```

Pour que votre script soit utilisable dans Apache il faut qu'il se trouve dans un répertoire particulier. Le répertoire

`$HOME/public_html/cgi-bin`

est configuré dans notre exemple pour accepter de lancer des scripts. Vous devez également vous assurer que votre script est exécutable avec la commande

`chmod 755 [script.sh]`.

Lorsqu'un client se connecte sur un serveur, celui-ci récupère du client l'adresse source, le type de client, qu'il passe au script CGI sous la forme de variables d'environnements. De la même manière, lorsqu'une ressource est demandée par la méthode GET elle peut

contenir des paramètres ajoutés à la fin du nom de la ressource. Le format de passage des paramètres est :

```
<a href="toto.cgi?param1=toto&param2=titi&param3=25">afficher</a>
```

La zone paramètre est séparée de la zone ressource par un ? Les différents paramètres sont séparés par des &. Tous les paramètres sont encodés au format 7 bits (pas de caractères spéciaux). Lorsque le serveur Web reçoit une requête GET avec une ressource de type CGI il extrait la chaîne située après le & et crée une variable d'environnement QUERY_STRING qui est passée au script. A la charge du script de traiter les paramètres.

la séparation & entre les paramètres n'est qu'une convention. La variable QUERY_STRING contient la chaîne en brut. L'ensemble des variables qui sont mises à disposition dans les scripts est accessible en écrivant un script shell utilisant la commande set.

```
#! /bin/sh
echo "Content-Type: text/html"
echo "" # ligne vide
echo "<html><body> "
echo `set`
echo "</body></html>"
```

NET– TD 6 : Annuaires et DNS

Questions :

1. A quoi peut servir un service de nommage ?
2. Quels sont les services de nommage connus ?

1 Réalisation d'un annuaire

Questions :

1. Proposez une solution centralisée (rédaction de l'annuaire par un responsable unique).
2. Pourquoi cette solution n'est pas bonne par rapport à ce que vous connaissez déjà de la structuration de l'Internet ?

2 Annuaire de grande taille

La mise en place d'un annuaire de grande taille est nécessairement distribuée. Chaque groupe possède une délégation de responsabilité pour les enregistrements dont il est responsable.

Questions :

1. Proposez une structuration d'un annuaire avec délégation.
2. Proposez une méthode pour rechercher l'information dans un annuaire.
3. Un annuaire est-il forcément hébergé parmi les machines qu'il est censé gérer. Formulé autrement : y a t'il un rapport entre la machine qui gère l'annuaire et les machines qui sont répertoriées dans l'annuaire ?

3 Annuaire inverse

Questions :

1. Comment pourriez vous réaliser un annuaire inverse avec le même fonctionnement que l'annuaire direct ?

Annexe B

Travaux pratiques

NET– TP 1 : Installation, configuration de réseaux IP, outils

1 Votre premier réseau : un cas simple

1.1 Mise en place

Vous devez construire deux réseaux IP de 2 machines chacun (voir schéma). Dans ce TP nous allons configurer des réseaux IP et voir l'interaction avec le niveau 2 (MAC) qui est ici un réseau Ethernet. Le premier montage doit pouvoir faire communiquer les machines 2 à 2 puis voir comment faire du routage pour passer sur plusieurs réseaux.

- Les 2 réseaux IP d'un même banc ne doivent pas avoir le même numéro de réseau.
- Vous brancherez tout de même les 4 machines sur le même commutateur Ethernet (les machines doivent dialoguer entre elles dans un même réseau IP mais les machines de deux réseaux différents ne peuvent pas communiquer entre elles).

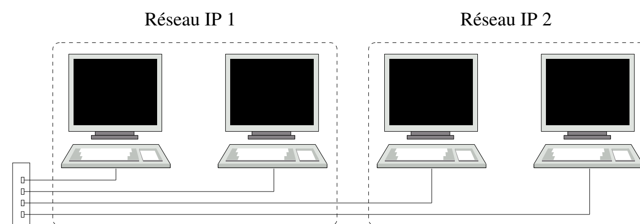


Figure B.1 – Branchement initial pour le TP (par banc)

Note pour se connecter aux machines

- Le login pour se connecter aux machines est "invite", le mot de passe associé est "linadm"
- Une fois que vous avez mis en place votre session, vous devez lancer une machine virtuelle pour effectuer le TP. Nous utiliserons la machine virtuelle "DEBIAN_LANWAN3" (aller dans le menu "system" puis "vmplayer"). Les machines virtuelles sont installées à la racine.
- Une fois la machine virtuelle lancée vous devez vous reconnecter mais avec le compte "root" cette fois. Le mot de passe est toujours "linadm".
- Les interfaces configurées sont "eth0" et "eth1". Ces cartes sont celles situées en bas des machines, eth0 est placée au dessus de la carte eth1.

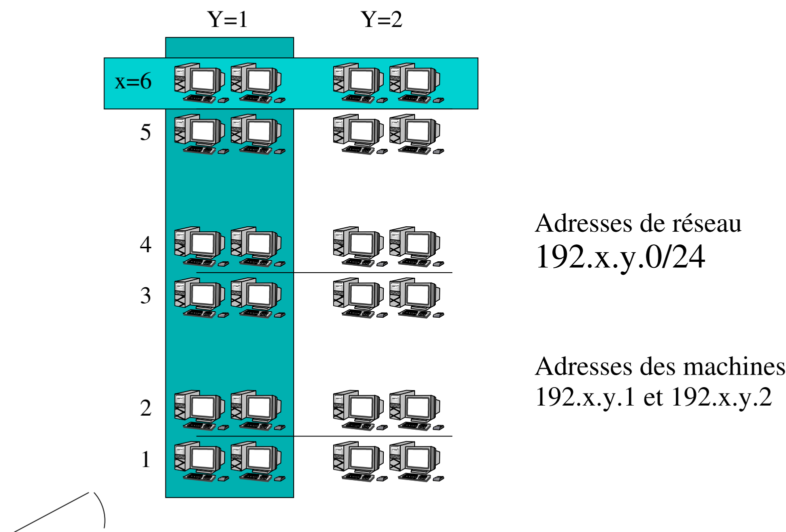


Figure B.2 – Proposition pour les numéros de réseau

Notes : en salle TP REZO, vous pourrez utiliser les switchs Cisco 2960 étiquetés de 7 à 12. Chacun de ces switchs dispose de 24 ports.

- Vous pourrez utiliser un switch 2960 par demi-banc (1 à 6). Les demi-bancs se font face. Sur chaque switch les ports 1 à 12 sont dans le VLAN 1 et les ports de 13 à 23 sont dans le VLAN 2.
- Le port 24 peut être utilisé pour surveiller l'ensemble des ports quelque soit le VLAN. Par exemple, il suffit de brancher l'interface Ethernet d'un PC sur le port 24 et avec wireshark on peut voir tout le trafic du switch.
- Tous les ports de ces 6 switchs sont fixés à la vitesse de 10Mb/s.
- Les switchs numérotés 1 à 6 seront utilisés en fin de TP pour avoir certaines interfaces à 100Mb/s

1.2 Configuration

Vous disposez de 4 machines par paillasse. Convenez de leur répartition en deux groupes. Chaque système doit posséder son paramétrage IP, adresse et masque. Pour la suite du TP il est fortement conseillé de coller au plan d'adressage proposé dans la figure au dessus. Paramétrez les systèmes. Utilisez les commandes comme `ifconfig` et `route` pour valider votre configuration.

La configuration d'un réseau IP se fait de la façon suivante :

`ifconfig` *eth0 192.x.y.num_machine* `netmask` *255.255.255.0*

Pendant que le réseau fonctionne, vous pouvez utiliser la commande `wireshark` (qui doit être lancée depuis un shell root) pour voir tout ce qui se passe sur le réseau.

La commande `ping [adresse IP]` permet d'envoyer un paquet à la machine destination et de demander un paquet en retour. Cela permet de tester si la connection est bonne et si toute la configuration jusqu'au niveau IP est correcte.

1.3 Résultats

1. Repérez vos adresses Ethernet avec la commande `ifconfig`
2. Quels paramètres avez-vous donné à chaque système hôte ?
3. Expliquez pourquoi la communication entre des machines dans des réseaux IP différents ne peut être effectuée *malgré la présence d'un réseau Ethernet commun.*
   – Vérifiez la présence de communications entre les machines d'un même réseau (`ping`)
   – Utilisez les captures de trame pour mettre en évidence les communications de type `ARP`.
   – Expliquez comment le destinataire d'un datagramme IP est trouvé par l'émetteur.
4. Essayez de mettre en évidence le fonctionnement de la résolution d'adresses avec 3 tests (vous avez besoin de l'outil de capture)
   – `ping` vers votre voisin
   – `ping` vers 192.x.y.3 (machine qui n'existe pas mais qui est dans votre réseau)
   – `ping` vers une machine du banc à côté de vous.
   Citez les tests que vous effectuez et les résultats commentés des commandes pour trouver comment fonctionne ARP.

2 Routage élémentaire

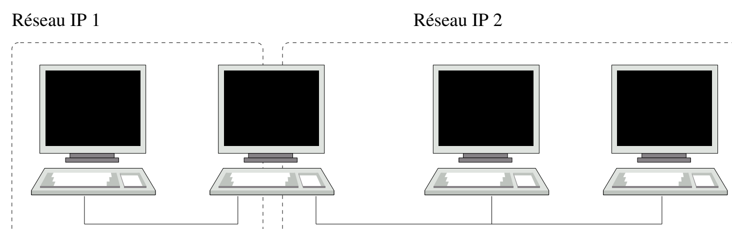


FIGURE B.3 – Configuration de routage sur un banc

2.1 Objectifs Une des machines doit être configurée au moyen de deux cartes ethernet pour appartenir aux deux réseaux. Cette machine va servir de routeur (aussi appelé “passerelle”) et permettre aux 4 machines de communiquer entre elles.

2.2 Mise en oeuvre Vous disposez de 4 machines par rangée. Convenez d'affecter à une de ces machines la fonction de routeur. Chaque machine doit posséder, dans son paramétrage IP, la connaissance de la façon d'atteindre toutes les machines du nouveau réseau. Paramétrez les systèmes en utilisant la commande `route`.

– `route add -net` *192.x.y.0* `netmask` *255.255.255.0* `gw` *@IP*

L'adresse `IP` à donner en argument aux commandes est l'adresse de la passerelle *dans votre réseau local*. Le routeur, comme il est connecté au 2 réseaux, n'a pas besoin qu'on lui indique d'autre information supplémentaire.

2.3 Résultats

1. Quels paramètres avez-vous donné à chaque système hôte ?
2. Expliquez comment le destinataire sur le réseau IP 2 d'un datagramme est trouvé par l'émetteur présent sur le réseau IP 1.
3. Vérifiez la présence de communications entre les systèmes. Citez les tests que vous effectuez.
4. Effectuez des captures de trafic sur le réseau pour retrouver toutes les trames échangées.

Remarque : la machine ayant deux cartes ethernet nécéssite une configuration spéciale pour agir en tant que routeur. Sans cette configuration les paquets lui seront bien délivrés mais la machine ne les réémettra pas (elle se comporte comme un PC qui serait simplement connecté à 2 réseaux). Pour activer la fonction de routage il faut sous Linux modifier la valeur de la variable du noyaux `ip_forward` :

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

3 Routage plus complexe

Cette partie doit permettre de mettre en place le routage sur la salle complète. Vous devez utiliser un nouveau switch pour connecter une machine par banc dans un nouveau réseau 193.0.0.x/24 avec x votre numéro de banc.

3.1 Réglage du routage

1. Quelles sont les règles de routage que vous avez rajouté sur les machines connectées au réseau 193/24 ?
2. Combien de règles avez vous du rentrer ? Auriez-vous pu diminuer le nombre de règles et si oui comment ?
3. Faites une boucle de routage volontaire et observez comment se comporte les paquets. Trouver la boucle la plus simple que vous puissiez mettre en place.

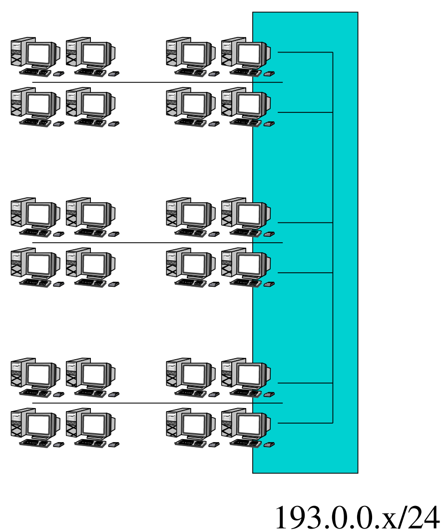


FIGURE B.4 – Configuration finale

4. Expliquer le fonctionnement de vous mettriez en place pour les annonces et la mise en place de configuration automatique des tables de routage.

NET– TP 2 : TCP et UDP

Note : Ce TP fait l'objet d'un compte rendu. Vous devrez rendre un rapport d'au maximum 5 pages sur les manipulations et mesures effectuées pendant le TP. La dernière question de ce TP doit être rédigée et détaillée dans votre compte-rendu. Vous devez rendre un compte rendu par banc (2 binômes).

1 Présentation

Le but de ce TP est de vous familiariser avec les analyses de traces des protocoles réseaux ainsi qu'avec des analyses de performances simples.

La première étape est de remonter la plate-forme du TP précédent en mettant en place le routage IP sur la machine présente sur les 2 réseaux Ethernet. Vous devez configurer un réseau IP par réseau local Ethernet. Pour ce montage vous ferez attention à mettre un réseau Ethernet à **100Mb/s** et un réseau ethernet à **10Mb/s**. Les switchs numérotés de 1 à 6 sont à 100Mb/s, ceux numérotés de 7 à 12 sont à 10Mb/s.

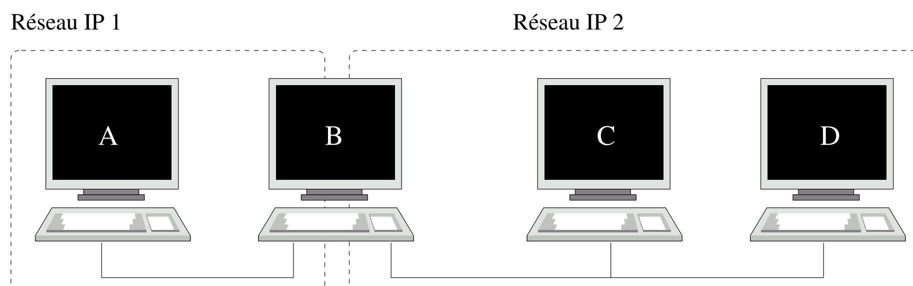


FIGURE B.5 – Montage du banc de test

2 Analyses et vérifications simples

Pour vérifier les performances des réseaux que vous venez de monter vous devez faire une première mesure de flux UDP et TCP entre les machines de votre réseau.

Vous devez mesurer le temps d'exécution (avec la commande `time`) d'un échange de données réalisé avec le logiciel `sock`. La mesure de la quantité de données reçues par le serveur doit être faite en effectuant une redirection de la sortie standard vers un fichier (et en enlevant le paramètre `-i` sur le serveur uniquement).

La documentation de `sock` est à la fin du sujet de TP !

Vous ferez les mesures entre les machines
- $A \to B$
- $A \to C$
- $C \to A$
- $C \to B$

1. Quels sont les débits que vous avez observés dans chacun des cas pour UDP et TCP ?
2. Expliquez vos résultats et justifiez chacun des débits.
3. Quel est le principal mécanisme mis en avant par cette vérification simple des débits ?
4. Estimez la taille maximale des buffers utilisés pour la pile réseau de la machine qui sert de routeur.
5. Justifiez votre réponse précédente en indiquant quelle est l'expérience qui permet de trouver la valeur.
6. Dans ce dernier cas, donnez une estimation précise de la séquence des paquets qui sont arrivés au serveur.

3 Analyse de performances

Montage : Vous devez monter un réseau avec 2 routeurs. Vous aurez donc au final 3 réseaux IP (A+B), (B+C) et (C+D) : la machine C est transformée en routeur. Les trafics générés pourront donc être entre deux réseaux locaux avec la possibilité de passer par un troisième réseau intermédiaire.

L'analyse de performance à l'aide de commandes simples (ping, sock) va permettre de retrouver certains paramètres du réseau que vous avez monté. Ces performances devront mettre en évidence les paramètres suivants :

3.1 Influence de la fragmentation

Faites varier la MTU des liens ethernet (à l'aide de la commande `ifconfig`) pour mesurer l'influence de celle-ci sur la fragmentation d'un paquet `IP`. Essayez également avec une connexion `TCP` et reporter les tailles de `MSS` annoncées lors de l'ouverture de session.

- Y a t'il de la fragmentation lorsque vous utilisez `TCP` entre A et C ? Justifiez.
- Mettez en évidence une mesure qui permet de mesurer le débit avec la prise en compte de la fragmentation. Justifiez.

3.2 Temps de latence du réseau

A l'aide des outils, dont `ping` et `netperf`, effectuez des mesures de temps de latence pour trouver la latence du réseau (paquet de taille 0). Comme il n'est pas possible d'envoyer un **paquet de taille 0** vous devez faire une série de mesure permettant d'**extrapoler ce temps de latence**.

1. Faites les mesures entre deux machines dans un même réseau local et donnez une estimation du temps de latence.
2. Faites la même série de mesures en traversant un routeur. Donner une estimation de la décomposition des temps lors du trajet du paquet (réseau, machine, routeur, etc...).
3. Justifiez votre réponse en traversant maintenant une deuxième routeur sur le trajet et en effectuant de nouvelles mesures.
4. Quelle est l'influence du débit du réseau dans le temps de latence. Justifiez et commentez par rapport aux résultats obtenus dans les précédentes mesures.

3.3 Mesure de débit utile

Mesurez les débits utiles obtenus pour les protocoles UDP et TCP en présence de fragmentation. Vous devrez mesurer les débits entre deux machines sur le même LAN, puis entre deux machines en utilisant du routage.

1. Tracez une courbe (taille de paquet IP, débit) en utilisant sock en UDP et son option `-i` pour générer du trafic. Essayez de mettre en avant l'influence de la MTU sur le débit (tâchez d'éviter à la perte des paquets pour ne pas fausser les mesures : faites des tests entre A et B et entre C et D).
2. Tracez une courbe (taille de MTU, débit) en utilisant `sock` dans le mode que vous voulez pour mettre en évidence l'influence de la MTU dans le débit lorsque la taille des paquets est constante.

3.4 Partage de connexion TCP

Faire maintenant un test de génération de trafic à l'aide de plusieurs machines sources vers une même destination.

1. Essayez avec deux sources vers une destination : quel est le débit moyen ?
2. Même question pour une configuration avec 3 sources.
3. Commentez en quelques lignes le fonctionnement et les résultats des mesures que vous venez d'effectuer.

4 Outils TCP/IP sous Linux

La syntaxe des commandes les plus utilisées pour l'administration de machines TCP/IP sous Linux sont indiquées ci-dessous. Un grand nombre de ces commandes est implanté dans d'autres OS comme Windows ou MacOS. Attention toutefois, les options ne sont pas forcément les mêmes ainsi que la présentation des résultats ! Bien entendu cette courte présentation ne saurait remplacer les pages du manuel (`man`).

La plupart des outils nécessitent d'être administrateur des machines pour faire des modifications. Vous pouvez tout de même les utiliser pour consulter les configurations en tant qu'utilisateur.

/bin/netstat Permet le listage de la table de routage des datagrammes IP et l'état du réseau.

Quelques options :
- `-a` : montre l'état de tous les sockets
- `-i` : montre l'état des interfaces (carte réseau, etc.)
- `-r` : montre les tables de routage
- `-s` : montre les statistiques pour chaque protocole
- `-n` : affiche les adresses réseau sous forme de numéros IP.

/bin/ping Envoi des paquets `ICMP ECHO_REQUEST` vers un système cible et réception d'un écho `ECHO_REPLY` retourné en réponse par le système cible. C'est donc un test élémentaire pour vérifier la bonne circulation des datagrammes IP dans un réseau.

`ping @IP-hôte-distant`

Quelques options :
- `-c num` : nombre de paquets à envoyer
- `-i sec` : défini le nombre de secondes entre l'envoi de deux paquets
- `-s taille` : taille des paquets à envoyer en octets
- `-n` : affiche le nom des hôtes cible sous forme de numéro IP
- `-R` : enregistre la route empruntée par les paquets
- `-v` : affiche le détail des paquets reçus

/usr/bin/traceroute Indique quels sont les routeurs traversés entre la machine d'origine et une machine destinataire dont on spécifié l'adresse IP. Peut servir de test pour la vérification de la bonne circulation des datagrammes IP dans un réseau :

```
traceroute @IP-hôte-distant
```

tcpdump Affiche une trace (capture) des datagrammes IP qui transitent sur une interface réseau.

```
tcpdump [options] expression
```

`-c num` : termine après avoir reçu le nombre de paquets spécifiés
`-i interface` : affiche le trafic présent sur l'interface précisée
`-n` : affiche les adresses réseau sous forme de numéro IP
`-t` : n'affiche pas d'informations sur la date d'émission du paquet
`-v` : procure plus d'informations (time to live, type of service, etc.)
`-x` : affiche l'en-tête sous forme hexadécimale
`expression` : restreint les paquets affichés à certains hôtes, ports ou protocoles. Consiste en une suite de primitives restreignant les paquets à afficher, chaque primitive étant : un identificateur (hôte, réseau, numéro de port) précédé de son type (host, net ou port) et éventuellement d'une direction (src, dst, src or dst ou src and dst) un protocole (ether, ip, arp, rarp, tcp, udp, ?)

Cet outil n'est pas utilisé dans la salle de TP info mais sera disponible dans la salle de TP réseau.

ethereal / wireshark Interface graphique de capture et de décodage de trames. Propose les mêmes fonctions que tcpdump mais avec une interface graphique qui permet de visualiser directement les différents protocoles utilisés. Cet outil n'est pas non plus installé dans les salles de TP info, il est par contre installé dans la salle de TP réseau.

4.1 Fichiers importants

La configuration des machines est en général stockée (sur les machines Unix) dans le répertoire `/etc`. Certains fichiers de ce répertoire concernent la configuration du réseau et sont standards :

- `/etc/hosts` : configuration statique de la traduction entre nom de machine et adresse IP
- `/etc/resolv.conf` : configuration du résolveur DNS
- `/etc/services` : table indiquant les noms des protocoles en fonction de leur numéro
- `/etc/protocols` : table indiquant les noms des ports connus ("well known ports")
- `/etc/host/conf` : choix de configuration entre résolution statique et DNS pour les noms de machine
- `/etc/hostname` : nom de la machine locale

Ces fichiers sont presque toujours présents sur une machine Unix (Linux, MacOS et même certaines versions de Windows). Des fichiers supplémentaires sont disponibles dans /etc/ pour indiquer les paramètres de configuration spécifiques à la machine. Ces fichiers de configurations sont spécifiques à la distribution Linux utilisée. Sur les distributions Debian, comme au département, le fichier principal de configuration est
`/etc/network/interfaces`.

4.2 Présentation de l'outil Netperf

Cette fiche donne les fonctionnalités de base de l'outil de benchmark de performance réseau nommé netperf. Ce programme permet de faire divers tests entre un PC client et un PC distant (aussi nommé serveur). Netperf est un outil client/serveur permettant d'injecter du trafic dans le réseau jusqu'à le saturer pour en mesurer les performances.

Architecture de netperf

Netperf suit une architecture très classique client/serveur et il y a deux exécutables : netperf et netserver. Quand netperf est exécuté, la première chose qu'il fait est d'établir une connexion TCP de contrôle vers le PC distant.

Configurer le serveur netperf

Il est possible d'exécuter netserver à la main sur le PC cible de votre choix : `netserver -p <port>` et il s'exécutera et répondra aux requêtes sur le port que vous spécifiez (Si le port spécifié est différent du port standard veillez à le spécifier par la suite dans tous vos appels à netperf).

Netperf pour mesurer un taux de transfert

L'utilisation la plus courante est de vouloir mesurer le taux de transfert de donnée en blocs, i.e., un flux unidirectionnel et ceci revient à mesurer à quelle vitesse un système est capable d'envoyer des données vers un autre système.

Test de performance pour un flux TCP

Le test le plus simple est effectué par : `netperf -H <remote host>` qui va effectuer un test de 10 seconde. Les différentes options seront par défaut et la taille des buffers des sockets du PC client et du PC serveur distant sera celle du système par défaut.

-s sizespec which will set the local send and receive socket buffer sizes to the value(s) specified. [Default : system default socket buffer sizes]
-S sizespec which behaves just like -s but for the remote system
-m value set the local send size to value bytes. [Default : local socket buffer size]
-M value which behaves like -m, setting the receive size for the remote system. [Default : remote receive socket buffer size]
-l value set the test length to value seconds when value is > 0 0 and to |value| bytes when value is < 0
-D set the TCP_NODELAY option to true on both systems
Et bien d'autres dans la documentation.

Test de performance pour un flux UDP

Un test de performance employant UDP est assez similaire à TCP. Une différence majeure est que la taille envoyée (send size) ne peut pas être plus grande que le plus petit buffer du PC client ou distant, i.e., si vous spécifiez l'option `-m`, la valeur doit être inférieure ou égale à la taille des buffer de socket spécifié par `-s` et `-S`. Le seconde remarque est que UDP n'étant pas le test par défaut, il faut préciser UDP_STREAM : `$NETHOME/netperf -H remotehost -t UDP_STREAM - -m 1024`

Petite remarque triviale : UDP n'étant pas un protocole fiable, il est toujours bon d'examiner les résultats en émission et en réception.

Netperf pour mesurer des temps de requête/réponse

Netperf envoie des transactions d'une taille donné qui sont en fait une seule requête et une seule réponse.

Test de performance pour requête/réponse via TCP

La commande
`$NETHOME/netperf -H remotehost -t TCP_RR`
va employer une requête de taille 1 octet et une réponse de taille 1 octet. Les options sont :

-r sizespec set the request and/or response sizes based on sizespec.
-l value set the test duration based on value. For value > 0, test duration will be value seconds. Otherwise, test duration will be |value| transactions.

-s sizespec which will set the local send and receive socket buffer sizes to the value(s) specified. [Default : system default socket buffer sizes]
-S sizespec which behaves just like -s but for the remote system
-D set the TCP_NODELAY option to true on both systems

Test de performance pour requête/réponse via UDP

Idem que pour TCP avec les mêmes options (sauf `-D`!). Vous pouvez employer la commande `$NETHOME/netperf -H remotehost -t UDP_RR`

4.3 Time

La commande time(1) permet de mesurer l'exécution d'un programme. On peut mesurer ainsi relativement précisément le temps que prend un transfert de données avec les différents protocoles.

4.4 Sock

Sock permet de générer du trafic UDP ou TCP entre un client et un serveur.

L'aide est obtenue en tapant simplement sock sur la ligne de commande. A titre d'exemple voici la façon de générer un trafic UDP saturant une ligne.

```
serveur : sock -u -s @port
client : sock -u -i -n 10240 @ip_serveur @port
```

le paramètre `-n` correspond à un nombre de paquets de 1024 octets chacun à envoyer sur le réseau (-n 10240 correspond à un envoi de 10Mo). Le paramètre `-i` à 2 interprétations :

- sur l'émetteur : il indique à sock de générer des données arbitraire à envoyer (sans ce paramètre le programme attend des données sur son entrée standard).
- sur le récepteur : il indique de détruire les données à leur réception. Il faut enlever le paramètre si l'on veut pouvoir rediriger la sortie vers un fichier pour mesurer la taille des données reçues.

4.5 Ftp

On peut automatiser un transfert ftp afin d'en mesurer le temps plus précisément : il faut définir un ficher de configuration /.netrc

Il faudra changer les droits en 600 sur ce fichier (chmod 600 .netrc) pour qu'il soit pris en compte par ftp. Exemple de fichier :

```
machine 192.168.31.100 login root password linadm
macdef init
put fichier
bye
<nl>
```

La première ligne permet d'automatiser la connexion à la machine 192.168.31.100. On définit, ensuite, une macro avec le nom spécial init qui sera exécutée dès la connexion établie. Il ne faut pas oublier la ligne vide en fin de fichier, c'est elle qui marque la fin de la macro.

Ftp donne aussi un temps de transfert, qui sera forcément différent de celui donné par `time`. Il ne faut pas oublier que `time` mesure aussi le chargement du programme, l'établissement de la connexion, l'envoi des commandes ainsi que la clôture de la connexion.

NET– TP 3 : Liaison modem et PPP

Note : Ce TP fait l'objet d'un compte rendu. Vous devrez rendre un rapport d'au maximum 5 pages sur les manipulations et mesures effectuées pendant le TP. La dernière question de ce TP doit être rédigée et détaillée dans votre compte-rendu. Vous devez rendre un compte rendu par banc (2 binômes).

1 Présentation

L'objectif du TP est de réaliser des connexions point à point entre client et serveur en utilisant le protocole standard ppp (rfc 1661). Ce TP est une version simplifiée des approches proposées pour les connexions aux fournisseurs d'accès à Internet.

1.1 Présentation de ppp

PPP (Point To Point Protocol) est un protocole négocié entre client et serveur. La négociation permet de définir d'une part les caractéristiques de la ligne physique et les caractéristiques de la couche réseau d'autre part avec des possibilités d'authentification et d'autoconfiguration pour le client.

Les critères sont négociés en deux phases :

- LCP : négociation de la ligne (RFC 1570).
- NCP : négociation du réseau. Lorsque cette négociation est faite pour un réseau IP on parle alors de IPCP (RFC 1332).

L'établissement d'une liaison spécialisée entre un client et un serveur se fait en utilisant une liaison point à point entre ces deux entités. Dans notre approche le serveur ppp attribue au client appelant une adresse IP qu'il possède sur le réseau. Puis le serveur se fait passer sur le réseau local pour la machine cliente concernant les trames ppp (usurpation d'identité arp).

1.2 Fonctionnement d'une connexion par modem

Prise en main des modems Pilotez les modems avec les outils (utiliser `minicom` sous Linux) et trouvez, en s'aidant de la documentation, les commandes permettant : de réinitialiser, de numéroter, de raccrocher, de reprendre la main sur une ligne établie, quitter l'outil sans raccrocher le modem, de demander à un modem de décrocher, de sauver une configuration de restaurer une configuration. (Attention aux fausses manipulations).

minicom : lancez minicom, tapez Ctrl-A + Z (contrôle + A puis Z, sans le contrôle). Allez dans le menu de configuration (touche

O), Réglage du port série puis réglage du débit (touche E) et enfin 115200 bauds (touche I). Ressortez du menu et choisissez `enregistrer config. sous dfl`.

Vous pouvez alors ressortir de minicom avec CtrA+Q et le relancer pour reconfigurer le port série avec vos nouveaux paramètres.

- CtrA+Z : page d'aide
- CtrA+Q : quitter minicom en laissant le modem configuré

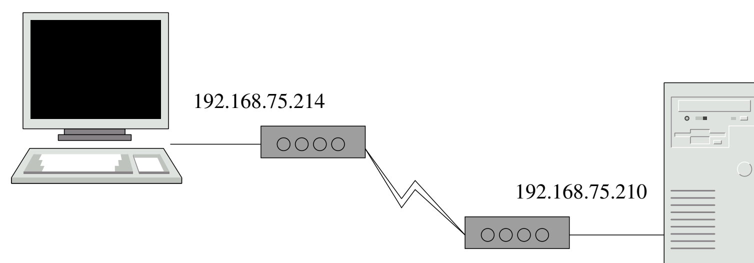
commandes du modem :

- ATZ : remise à zéro
- ATDT *numéro* : numéroter
- ATS0=n : décrocher après n sonneries

Une fois que vous aurez commencé à bien comprendre le fonctionnement des modems vous pouvez établir une liaison entre un modem client et un modem serveur sur le banc. Raccrocher et rappeler. Cette connexion est établie lorsque les deux modems se sont entendus sur les paramètres de base. Vous pourrez vérifier en restant dans `minicom` que la liaison, sans autre configuration logicielle, permet de faire passer des caractères tapés au clavier et les affiche dans la fenêtre minicom distante. Cette liaison bidirectionnelle servira de base aux montages suivants.

2 Test d'une liaison PPP simple

Nous allons dans un premier temps monter une liaison point à point simple permettant d'échanger des informations entre 2 machines comme dans la partie précédente mais au lieu d'échanger des caractères tapés au clavier ce seront les programmes pppd qui pourront s'échanger des informations et mettre en place un réseau IP.



Pour mettre en place votre première connexion vous devez piloter les modems pour établir la connexion entre les machines (une machine numérote et appelle l'autre, la machine qui reçoit l'appel se contente de décrocher). Une fois la connexion établie, assurez vous que la liaison en mode caractère fonctionne bien. Vous devez alors quitter `minicom` **sans réinitialiser les modems** (`Ctrl-A` puis `Q`),

sur une seule des deux machines. Une fois revenus sur la ligne de commande vous pourrez lancer le démon pppd à la main avec la ligne de commande suivante :

```
pppd /dev/ttyS0
```

Assurez vous que sur la machine qui est restée avec minicom vous avez bien la réception des données envoyées par pppd.

Question : Que constatez vous si vous exécutez le programme ifconfig -a sur la machine qui a lancé pppd ?

3 Réalisation d'une liaison PPP client-serveur

Ce type de configuration est celui rencontré lorsqu'on se connecte par modem à un fournisseur d'accès à Internet (FAI ou *ISP Internet Service Provider*). Le serveur est la machine recevant les appels et sert à un client appelant à se connecter à Internet. Le démon ppp est donc en attente permanente de demande de connexion et le modem associé doit être configuré pour décrocher automatiquement et laisser la main à pppd pour qu'il puisse envoyer les paramètres de configuration au client.

3.1 Configuration du serveur

Installer le modem sur le serveur Linux afin qu'il réponde aux appels et qu'ils attribuent une adresse réseau au client. Pour cela le démon pppd utilise les fichiers suivants :

```
/etc/ppp/options
/etc/ppp/options.ttyS0
```

Les options courantes pour le fichier options sont les suivantes :

```
asyncmap 0
lock
crtscts
modem
debug
kdebug 5
noauth
```

Les options `debug` et `kdebug 5` permettent de récupérer les traces de debug issues des programmes.
Les traces d'exécution passent par un démon (syslogd) qui se charge à partir d'un fichier de configuration d'écrire dans des fichiers adaptés les traces générés par les démons (où les simples exécutables). Dans le cas de ppp il faut lancer le programme avec la commande

```
/etc/init.d/sysklogd restart
```

Les fichiers de log sont
- `/var/log/messages`
- `/var/log/debug`.

Le fichier `options.ttyS0` permet d'associer chaque sortie série à une liste d'options spécifiques à la ligne. Un exemple d'options pour le fichier d'option de ligne `options.ttyS0` sont :

```
192.168.75.210:192.168.75.214
# adresse_locale:adresse_distante
```

Enfin, lancez le serveur sous linux afin qu'il soit à l'écoute du port série :

```
pppd /dev/ttyS0
```

Question : Une fois la connexion établie. Donnez les tables de routage constituant la configuration de la communication sur chacune des machines (route -n). Comment fonctionne l'interaction entre le niveau IP et le niveau PPP ?

Question : Donnez sur un schéma temporel les échanges de trames entre client et serveur ?

3.2 Configuration du client

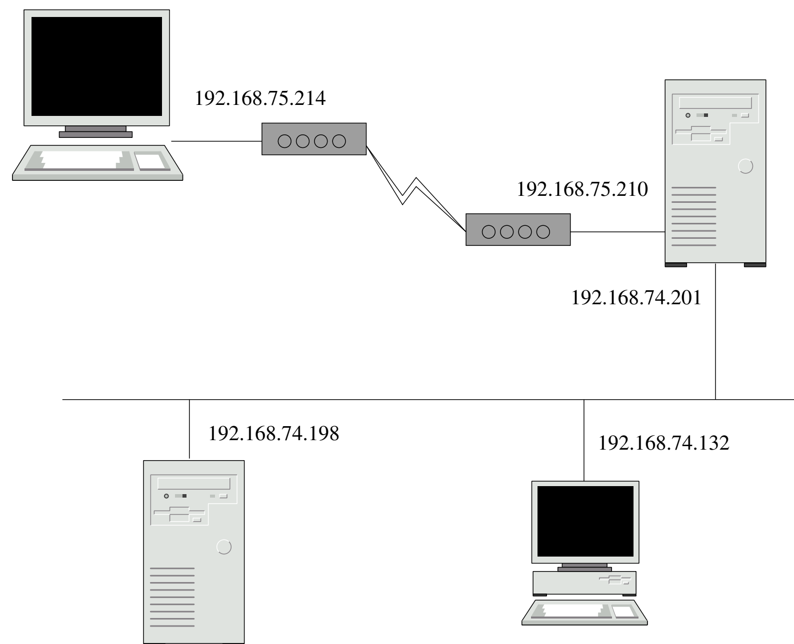
Dans le cas d'un client Linux, comme pour Windows, l'approche est la même. Le client doit d'abords établir la connexion téléphonique puis reprendre la main avec la mise en place du protocole ppp.

Question : Essayez de vous connecter sur le serveur et analyser la réponse de celui-ci.

Question : Une fois la connexion avec le serveur établie, quitter l'application de contrôle du modem *sans le raccrocher* et lancer le programme `pppd` sur la ligne série du modem. Observer les informations envoyées dans les fichiers de log.

4 Configuration et raccord sur un réseau local

Réaliser le montage suivant à l'aide de la liaison modem et de la configuration du premier TP. Dans un premier temps la liaison modem et la liaison Ethernet ne sont pas dans le même réseau local.



Question : Mettez en place le réseau sur la ligne modem et les tables de routage appropriées.

Question : Essayez de mettre en place un mécanisme de `proxyarp` sur la machine serveur de connexion. (voir la page du manuel de `pppd`)

Question : A quoi sert ce mécanisme et comment fonctionne t'il ? Que faut'il changer dans les configurations des machines.

5 Questions sur le mode de communication point à point

Le mode de communication qui est proposé dans ce TP est un mode de communication point à point utilisant une connexion série.

Ce mode de communication n'est pourtant pas limité à une utilisation simple comme nous venons de le voir.

N'importe quel moyen de communication peut être utilisé comme liaison de base pour mettre en place ppp à condition qu'elle soit bidirectionelle. On peut donc mettre en place une liaison ppp sur une liaison `telnet` ou `ssh`.

Proposez une utilisation d'une liaison ppp utilisant une liaison ssh comme support.

Questions :

1. Quelles sont les piles de protocoles mises en œuvre aux pour la communication ?
2. Détaillez les datagrammes en transit sur le réseau.
3. Quelles sont les adresse IP visibles sur le réseau ?
4. Décrire les tables de routage utilisées par le machines ?

NET– TP 4 : Programmation d'un serveur web (simple)

1 Introduction

HTTP est un protocole de niveau application spécifiquement conçu pour l'échange de fichiers hypertexte. Deux versions ont été normalisées à ce jour HTTP 1.0 [RFC 1945] et HTTP 1.1 [RFC2616]. la version1.1 offre une compatibilité ascendante avec la version1.0 en y ajoutant de nombreuses fonctionnalités. Dans le cadre de ce TP nous nous concentrerons sur les fonctionnalités fondamentales de la version 1.0

On se propose d'adopter, pour la réalisation de ce serveur, une démarche ajoutant progressivement des fonctionnalités à un squelette de base. Toutefois, pour ne pas avoir à réécrire tout le programme à chaque nouvelle fonctionnalité ajoutée, il est important d'avoir bien en tête dès le départ l'ensemble des fonctionnalités à implémenter dans le serveur final et pour cela il est vivement recommandé de lire cet énoncé plusieurs fois et de procéder à une étape de conception préliminaire.

2 Serveur TCP séquentiel

Au moyen de l'API socket, construisez un premier squelette de serveur TCP séquentiel. Constitué d'une boucle sans fin, ce serveur se placera en attente de connexion et répercutera sur la sortie standard l'adresse IP du client et tout ce qu'il reçoit jusqu'à ce que ce dernier termine la connexion. Vous utiliserez pour cela les fonctions :

- `socket()` pour la création d'un socket serveur TCP
- `bind()` pour lier ce socket à un numéro de port sur votre machine (utilisez 8080)
- `listen()` pour déclarer le socket comme socket serveur
- `accept()` pour bloquer le programme en attente de connexion

Attention : utilisez les fonctions `htons()` et `htonl()` pour toutes les conversions d'adresses dans les structures `sockaddr`.

Attention : toutes les fins de ligne dans les entêtes doivent impérativement être codées par la séquence `CR LF` (*Carriage Return/Line Feed*, ASCII 13,10), il est déconseillé d'utiliser la séquence d'échappement `\n` dont la traduction ASCII est variable d'une plate-forme à l'autre.

Question : Pouvez vous expliquer l'utilité des fonctions `htons()` et `htonl()`.

Question : Testez votre serveur en utilisant l'utilitaire telnet comme client. Invoquez : `telnet localhost 8080`.

3 Serveur HTTP séquentiel

A partir du squelette précédent il est assez simple d'implémenter les premières fonctionnalités d'un serveur HTTP. Le travail consiste à interpréter la requête puis à formuler une réponse. Le RFC 1945 contient la description des formats utilisés. Il est vivement conseillé de vous y référer pour une description exhaustive.

Implémentez le traitement des requêtes de type GET en ignorant les entêtes additionnels et en ne considérant que deux types de réponses ; envoi d'un fichier HTML ou bien envoi d'erreur.

4 Serveur parallèle

Le serveur séquentiel précédent ne peut répondre qu'à une seule requête à la fois. On se propose de dépasser cette limitation en implémentant un serveur parallèle. Pour cela on utilise l'appel fork() qui permet de créer un nouveau processus fils du processus courant. Le processus fils partage les descripteurs de fichiers de son père et peut donc travailler avec les socket créés par le père.

Important : ne pas oublier de fermer dans le père, la socket retournée par `accept()`.

5 Gestion des types et des erreurs

Complétez le serveur pour gérer d'autres types de fichiers : notamment GIF et JPEG.

Compléter votre serveur pour ajouter la gestion des types MIME (types de fichiers que vous renvoyez) ainsi que la gestion des erreurs (distinguez les erreurs 400, 403, 404 et 500). Vous pourrez également ajouter la gestion des scripts CGI si le temps le permet.

NET– TP 5 : Applications et protocoles

1 Introduction

Ce TP permet d'illustrer la programmation d'une application simple. Nous allons faire une requêt DNS et lire la réponse. Une requête DNS est forgée dans un paquet UDP et envoyée à un serveur sur son port 53. La réponse sera renvoyée par le serveur sur le port qui a servi à envoyer la requête.

Requêtes DNS / UDP le format de requête pour les paquets DNS est précisé dans la RFC 1035. Des informations sont également présentes dans les notes de cours. Le format de l'entête des paquets est le suivant :

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Pour poser une question seules les entêtes (Header) et questions sont nécessaires. Pour le TP vous pouvez vous contenter de faire des requêtes avec l'entête et une seule questions de résolution d'adresse.

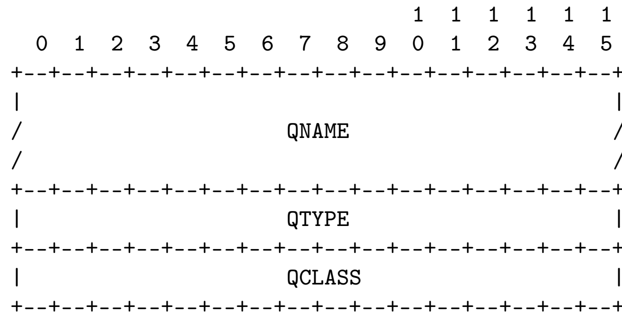
Format des entêtes

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

- QR : A one bit field that specifies whether this message is a query (0), or a response (1).
- OPCODE : A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are :
  - 0 a standard query (QUERY)
  - 1 an inverse query (IQUERY)
  - 2 a server status request (STATUS)
  - 3-15 reserved for future use
- AA : Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.
  Note that the contents of the answer section may have multiple owner names because of aliases. The AA bit
  corresponds to the name which matches the query name, or the first owner name in the answer section.
- TC : TrunCation - specifies that this message was truncated due to length greater than that permitted on the transmission channel.
- RD : Recursion Desired - this bit may be set in a query and is copied into the response. If RD is set, it directs the name server to pursue the query recursively. Recursive query support is optional.
- RA : Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server.
- Z : Reserved for future use. Must be zero in all queries and responses.
- RCODE : Response code - this 4 bit field is set as part of responses. The values have the following interpretation :
  - 0 : No error condition
  - 1 : Format error -
  - 2 : Server failure -
  - 3 : Name Error -
  - 4 : Not Implemented -
  - 5 : Refused -
  - 6-15 : Reserved for future use.

Pour un question standard tous les bits seront à 0 sauf le bit `RD` qui indique au serveur DNS que l'on souhaite une réponse finale sur la correspondance entre nom et adresse IP. Les champs QTYPE et QCLASS sont égaux à 1 (QTYPE = A : recherche d'adresse) et (QCLASS=IN : Internet)

Format d'une Question

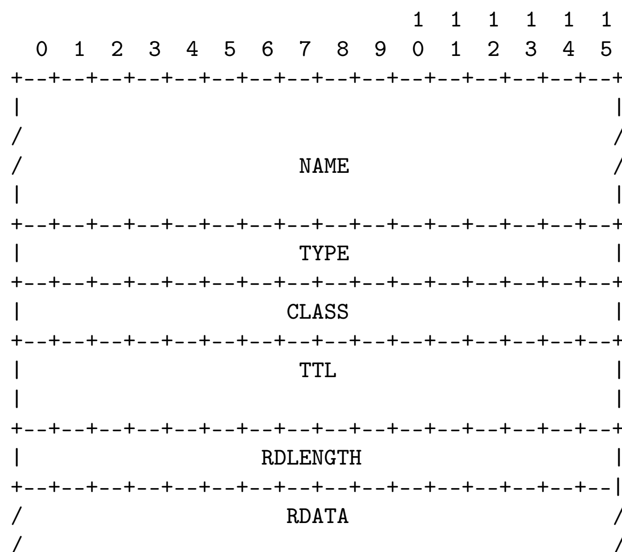


```
                                 1  1  1  1  1  1
   0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

– QNAME : a domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. Note that this field may be an odd number of octets ; no padding is used.
  QTYPE : a two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.
  QCLASS : a two octet code that specifies the class of the query. For example, the QCLASS field is IN for the Internet.

Le champ `QNAME` dans une question est celui pour lequel vous devez faire la transformation indiquée dans le code source pour le nom recherché (suppression des '.' et remplacement par la longueur du mot suivant).

Format d'une réponse (Answer)



```
                                 1  1  1  1  1  1
   0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
```

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

- NAME : a domain name to which this resource record pertains.
- TYPE : two octets containing one of the RR type codes. This field specifies the meaning of the data in the RDATA field.
- CLASS : two octets which specify the class of the data in the RDATA field.
- TTL : a 32 bit unsigned integer that specifies the time interval (in seconds) that the resource record may be cached before it should be discarded. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached.
- RDLENGTH : an unsigned 16 bit integer that specifies the length in octets of the RDATA field.
- RDATA : a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record. For example, the if the TYPE is A and the CLASS is IN, the RDATA field is a 4 octet ARPA Internet address.

2 Programmation et travail à effectuer

La construction d'une interrogation se déroule en plusieurs parties.

1. Créer une socket UDP (SOCK_DGRAM)
2. former un paquet de requête à partir d'un nom de machine
3. envoyer le paquet sur la socket au serveur DNS sur son port 53.
4. faire une lecture bloquante sur la socket pour la réponse.
5. décoder la réponse pour retrouver l'adresse IP renvoyée.

3 Où trouver des informations

- Vous pouvez prendre un début de programme sur Moodle et partir des sources pour compléter le programme.
- Les parties en anglais sont tirées du texte de référence de la définition du protocole qui est la RFC 1035. Le texte de la RFC est disponible ici :
  http ://www.ietf.org/rfc/rfc1035.txt
  http ://www.faqs.org/rfcs/rfc1035.html
- des notes sur les DNS et sur la programmation sont disponibles dans le supports de cours.